Improving DINOv2 Monocular Metric Depth Estimates by Smoothing Over Consecutive Frames

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

Pratik Chaudhari
University of Pennsylvania
pratikac@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 decoder. 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, as computed by phase correlation, to make the task of aligning and subsequently combining the features easier. The resulting set of DINO features are input to the downstream depth decoder, which produces the output depth map. We find that our approach, as applied to DINOv2 with the "small" encoder size, achieves a 23.8 percent reduction in MSE on the NYUv2 Depth dataset, resulting in a model which is both faster and more accurate than DINOv2 with the "base" encoder size. 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 with our approach, the output depth map is more accurately scaled to the ground truth. We believe that this occurs because our CNN-based adapter effectively smooths over the errors in depth estimation made across consecutive frames. We attribute the observed reductions in MSE primarily to this improved scaling. See our code at: **https://github.com/erikjagnandan/SmoothDINOv2**

II. Introduction

The application of large-scale transformer models in computer vision has led to the development of vision foundation models that demonstrate strong, generalized performance across vision tasks [1]–[3]. One such model is DINOv2, a vision transformer introduced by Meta AI, which has achieved state-of-the-art performance (among self-supervised models) for a variety of computer vision tasks, including depth estimation, semantic segmentation, and object detection. In order to train DINOv2, the authors first pretrain the ViT-based DINOv2 encoder via knowledge distillation, enabling the encoder to produce semantically-rich embeddings of input images. Then, a task-specific decoder model is placed downstream of the DINOv2 encoder, and is trained in a supervised manner to map the DINO features produced by the encoder to the predicted outputs for the desired task (e.g. depth maps). Notably, the authors develop separate DINOv2 pipelines—DINOv2-small, DINOv2-base, DINOv2-large, and DINOv2-giant—with varying encoder sizes, offering a tradeoff between performance and speed [1]. A basic visualization of the DINOv2 depth estimation pipeline is provided in Figure 1.

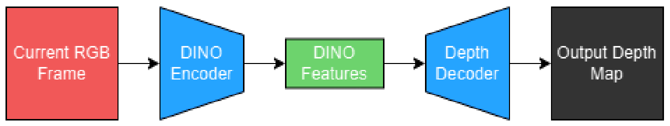

Fig. 1: DINOv2 Depth Estimation Pipeline

In this paper, we examine DINOv2's ability to perform monocular metric depth estimation on video stream data. We consider that, in video streams taken at real-time speeds (25-30 FPS), there exists strong temporal continuity between consecutive frames, as most parts of the environment seen at one frame are also visible in the previous frames. Thus, past frames provide relevant context regarding what is seen at the current frame, which can potentially be utilized to improve performance on vision tasks, such as depth estimation. However, we observe that DINOv2 does not leverage this continuity, and instead predicts depth independently for each frame without considering past frames. We propose to enhance the accuracy of the DINOv2 depth estimation pipeline by introducing a means of conditioning on past frames. Specifically, when objects are seen across several consecutive frames, and thus DINOv2 provides several independent predictions of their depth, 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 error than any individual depth estimate.

Our broad goal is to develop an approach which outperforms the existing DINOv2 pipeline in both speed and accuracy at a specific encoder size. We excluded the large and giant models from our analysis after finding they incur inference times over 500 ms on an NVIDIA T4 GPU, making them impractical for real-time use. Since our approach strictly introduces additional computation, it cannot improve upon the speed of DINOv2-small. Therefore, we set a specific goal of building an approach upon DINOv2-small which provides both faster and more accurate performance than DINOv2-base. Accordingly, all of our experiments apply our methods on DINOv2-small. Additionally, all of our experiments use the DPT decoder, the highest performing depth decoder from the original DINOv2 paper [1]. We quantify accuracy using the mean squared error (MSE), defined as $L_{MSE} = \frac{1}{NHW}\Sigma_{n=0}^{N-1}\Sigma_{i=0}^{H-1}\Sigma_{j=0}^{W-1}(d_{nij} - \hat{d}_{nij})^2$, where N is the number of data samples, H and W are the height and width of the depth map (which are equal to the height and width of the input RGB image), and d_{nij} and $\hat{d}_{nij}$ are the ground truth and predicted output depth for data sample n at pixel (i, j). In each of our experiments, we use MSE both as the loss function for training our models and as the evaluation metric for accuracy.

We summarize our contributions as follows:

1) Identifying that the errors made by the DINOv2 depth estimation pipeline can be reduced by combining the DINO features from consecutive frames
2) Extending our approach by inserting a small CNN between the DINOv2 encoder and depth decoder 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 to DINOv2-small resulted in a 23.8 percent reduction in the MSE, resulting in a faster and more accurate pipeline than vanilla DINOv2-base
3) Improving the resolution of the depth maps produced by our CNN-based pipeline by incorporating regularization into its training objective which encourages the updated set of DINO features produced by our CNN adapter to be similar to the original set of DINO features for the current frame. This improvement in resolution was met with only a small loss in accuracy, as the regularized CNN-based pipeline still achieved a 23.6 percent reduction in the MSE incurred by DINOv2-small

III. Related Work

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 aims to improve upon the robustness of depth estimation models by incorporating training data from a variety of datasets, thereby resulting in a model which performs well at various depth ranges and scales. While its original implementation relied on a convolutional-based design, more recent iterations of MiDaS have utilized vision transformers such as BEiT, Swin, and Next-ViT as their backbone encoder [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 targeted improvements specifically in metric depth estimation. For example, ZeroDepth employs a large transformer-based encoder-decoder architecture, achieving strong results across several depth datasets including KITTI (autonomous driving scenes) and NYUv2 (indoor scenes) [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, we aim 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.

IV. 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.

V. Approach

A. Dataset

The NYUv2 Depth dataset contains images of indoor scenes with corresponding depth maps. Its train split consists of

~400k images, but due to memory limitations, we select 50k images for our work. These training images, recorded from video streams in various indoor environments like bedrooms, basements, and cafes, are provided in order, allowing us to apply our approach to combine consecutive frames. However, the test split consists of unordered frames from different scenes, making it impossible to identify consecutive frames and apply our method to the test data.

To resolve this, we treat our 50k frames from the train split from NYUv2 Depth as our entire dataset, and perform a training-validation split in which we allocate ~40k image frames to the training set and ~10k image frames to the validation set.

B. *Hardware Details*

To accelerate the training process, all training was performed on a single NVIDIA GeForce RTX 4090 GPU with ~25 GB of RAM. However, all evaluation (notably including speed evaluation) was performed on an NVIDIA T4 GPU with ~15 GB of RAM. We chose to evaluate on the T4 GPU because a critical application of depth estimation is in robotics, where weight and cost limitations often restrict hardware options. As a result, evaluating on the more affordable and lightweight T4 GPU provides a more realistic estimate of the performance that could be seen on actual robots.

C. *Method*

In order to combine information from consecutive video frames, we use a small model placed in between the DINO encoder and depth decoder which produces a single set of improved DINO features given the DINO features from the current and previous frames. Since the original DINO features from 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 that propagates the original DINO features of the current frame to the output of the CNN. Thus, at the beginning of training, the CNN outputs a tensor of all zeros, which is added to the original DINO features of the current frame. Thus, the same original DINO features are input to the depth decoder. As training progresses, the model learns to adjust the features to improve depth estimation performance. We refer to this CNN placed between the DINO encoder and depth decoder as the "adapter model" or "adapter."

Note that if the adapter 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 adapter 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 the adapter. The DINO features for a single frame are of shape H/patch size $\times$ W/patch size $\times$ channel_dim, so we provide this additional input by creating two feature maps of shape H/patch size $\times$ W/patch size, one in which every value is equal to the horizontal shift and one in which every value is equal to the vertical shift. We then concatenate 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 adapter model, a 2-layer CNN with 3×3 kernels and a stride of 1, processes this input and produces an output of shape H/patch size $\times$ W/patch size $\times$ channel_dim. This output is added to the original DINO features via a skip connection and fed into the depth decoder. A graphic of our model architecture is provided in Figure 2. We trained our model for 10 epochs across our entire 40k image training set using the MSE loss, which is defined as $L_{MSE} = \frac{1}{NHW}\Sigma_{n=0}^{N-1}\Sigma_{i=0}^{H-1}\Sigma_{j=0}^{W-1}(d_{nij} - \hat{d}_{nij})^2$ in the Introduction section.

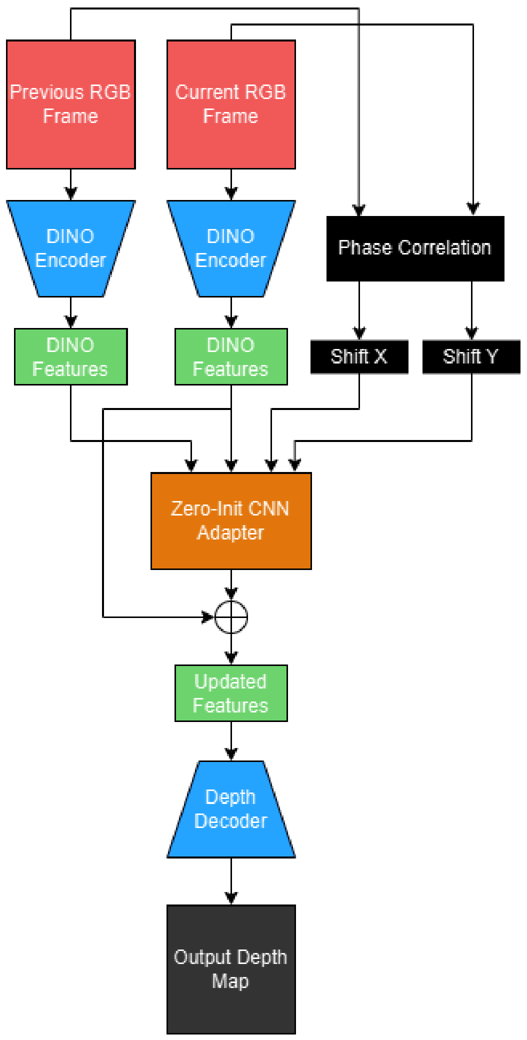

Fig. 2: Model Architecture

As seen in the Results section, applying our CNN-based adapter as described above achieves a substantial reduction in MSE, but also causes a degradation in the resolution of the output depth maps. To address this resolution loss, we introduce regularization to the adapter's training objective. This regularization encourages the updated DINO features to remain similar to the original features for the current frame. The modified loss function is $L_{MSE,reg} = \frac{1}{NHW}\Sigma_{n=0}^{N-1}\Sigma_{i=0}^{H-1}\Sigma_{j=0}^{W-1}(d_{nij} - \hat{d}_{nij})^2 + \lambda\frac{1}{NHWC}\Sigma_{n=0}^{N-1}\Sigma_{i=0}^{H_F-1}\Sigma_{j=0}^{W_F-1}\Sigma_{k=0}^{C-1}(f_{nijk} - \hat{f}_{nijk})^2$, where f_{nijk} and $\hat{f}_{nijk}$ are the original and updated DINO features for data sample n at index (i, j, k), F_H, F_W, and C are the height, width, and channel dimension of the DINO feature space, and all other terms are as defined in the Introduction section. Through manual tuning, we select $\lambda = 0.1$ as the optimal coefficient for the regularization.

D. Results

Our CNN-based pipeline without regularization achieves an MSE of 0.1421, a $\sim$23.8 percent reduction in MSE relative to vanilla DINOv2-small, while incurring $\sim$80 ms of additional latency. Notably, this performance exceeds that of vanilla DINOv2-base, constituting a $\sim$3.9 percent reduction in MSE relative to this model while also providing $\sim$20 ms faster inference. Each epoch lasted $\sim$1 hour, so the entire training procedure took $\sim$10 hours total. We provide the plot of the inference time versus MSE of each model in Figure 3.

While our approach significantly reduces MSE, examination of the output depth maps reveals a loss in resolution, as the updated depth map is considerably lower in resolution than the original. We believe this occurs because the MSE objective prioritizes large errors in depth scale but only weakly encourages the preservation of high-resolution details. For instance, an error of 1 meter in estimating the distance to a wall results in a large MSE of $1^2 = 1$, while a small error (e.g., missing a small object and instead predicting the depth to the wall 0.1 meters behind it) results in a small MSE of $0.1^2 = 0.01$, despite being important for resolution. As a result, the MSE objective drives the CNN-based adapter to rescale depth maps, neglecting high-resolution features.

Figures 18 and 19 in the Appendix show that the regularized pipeline produces depth maps with resolution comparable to those from vanilla DINOv2-small, a significant improvement over the unregularized pipeline. This enhancement comes with minimal impact on MSE performance, achieving an MSE of 0.1425—a $\sim$23.6% reduction relative to vanilla DINOv2-small, only slightly worse than the $\sim$23.8% reduction achieved by the unregularized pipeline. We do not include an updated inference time versus MSE graph, as the MSE difference between the regularized and unregularized pipeline is negligible and the inference time is identical; the points would overlap on the graph.

We use NCUT [8] to visualize how the adapter model, both with and without regularization, alters the DINO features output by the DINO encoder. Figures 7 and 8 in the Appendix show NCUT visualizations of the DINO features for the same input image (Figure 6) before and after modification by the unregularized adapter. Figures 13 and 14 display the corresponding NCUT visualizations for the regularized adapter. Relative to the original DINO features, the modified features produced by the regularized and unregularized adapter both show negligible change in the eigenvectors but significant changes in the eigenvalues. This indicates that the adapter

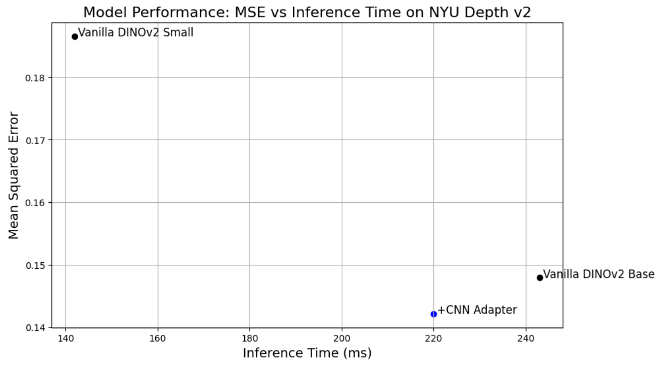

Fig. 3: Inference Time for One Image versus MSE for Each Approach

modifies the scale rather than the structure of the features, regardless of whether regularization is applied. The output depth maps for vanilla DINOv2-small and DINOv2-small with our adapter are shown in Figures 9 and 10 for the unregularized adapter and in Figures 15 and 16 for the regularized adapter. Similarly, both with and without regularization, we observe that the overall structure is consistent, but the scale of the updated depth map aligns more closely with the ground truth, as indicated by the colorbar. This suggests that our approach achieves its reduction in MSE by combining the information from consecutive frames to make the output depth maps more accurately scaled on average.

VI. Conclusions and Future Work

Our work demonstrates that the DINOv2 depth estimator can be improved by combining depth estimates from consecutive frames to reduce per-frame errors. Specifically, we achieve this by introducing a zero-initialized CNN between the DINO encoder and depth decoder, which computes refined DINO features for two consecutive frames based on their pixel shift from phase correlation. Regularizing these features to remain similar to the original frame's features enhances depth map resolution significantly, nearing DINOv2-small's resolution while negligibly sacrificing MSE performance.

We suspect that our approach can be applied to many vision models beyond DINOv2 which process video streams but lack mechanisms such as recurrence or conditioning to leverage temporal context. Moreover, since our adapter model operates in the latent space, it is task-agnostic and compatible with any downstream decoder. For example, we could replace the depth decoder with the DINOv2 decoders for semantic segmentation or object detection, and potentially improve performance on those tasks. This adaptability suggests broad potential for improving various models and modalities. However, a key limitation is that our approach relies on the model making varied errors across frames. If the model produces consistent errors in its estimates, then combining the depth estimates using our approach will not lead to any improvement.

Beyond exploring its applications to other models and modalities, we identify several additional opportunities for future work. For example, we only considered conditioning our adapter model on the current and previous frames, and further improvement could likely be obtained by conditioning on larger numbers of past frames. Additionally, the code for our implementation could be further optimized to provide faster performance. However, doing so would cause the input to the adapter to increase in dimension by channel_dim+2 for every additional past image (for the DINO features themselves, plus the horizontal and vertical shifts computed via phase correlation). As a result, computation would scale at $O(n)$ as more past frames were added, making our approach slower and less useful in real-time applications. To address this, we suggest replacing the CNN-based adapter with an RNN or LSTM which maintains a hidden state encapsulating the DINO features of the past frames and combines it with the DINO features of the present frame to produce an improved set of features. Such an approach would incorporate information from an arbitrary number of past frames with $O(1)$ computation, making it suitable to real-time usage.

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. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[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.

[8] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

Appendix

A. Model Performance Summary

The performance of each model, in terms of MSE and inference time, is summarized in Table 1.

Model	MSE	Inference Time (ms)
Vanilla DINOv2-Small	0.1865	142
+ CNN Adapter	0.1421	220
+ CNN Adapter Regularized	0.1425	220
Vanilla DINOv2-Base	0.1479	243

TABLE I: Comparison of Models Based on MSE and Inference Time

B. Training Curves

The validation MSE curves are shown in Figures 4-5, with Vanilla DINOv2-small and Vanilla DINOv2-base included as reference points. Since our approaches build on DINOv2-small, they begin with the same MSE, as the the zero-initialization of the adapter layer and its skip connection force the model to initially replicate the Vanilla model's features and output depth map.

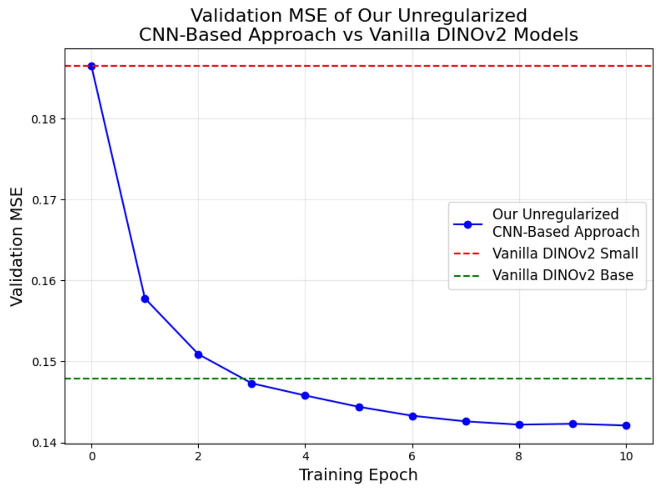

Fig. 4: Validation MSE Curve - Unregularized CNN Adapter

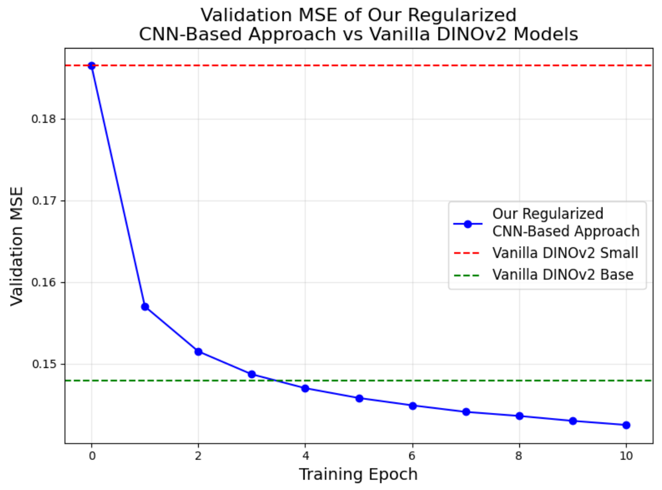

Fig. 5: Validation MSE Curve - Regularized CNN Adapter

C. NCUT Visualizations - Unregularized CNN Adapter

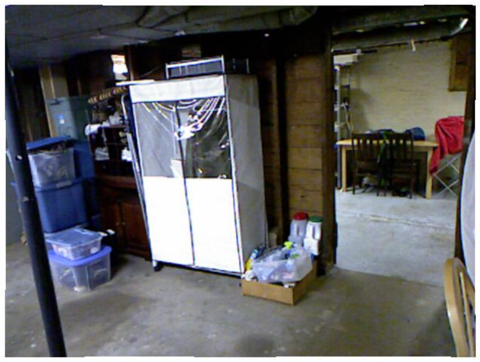
Fig. 6: Input RGB Image from NYUv2 Depth Dataset

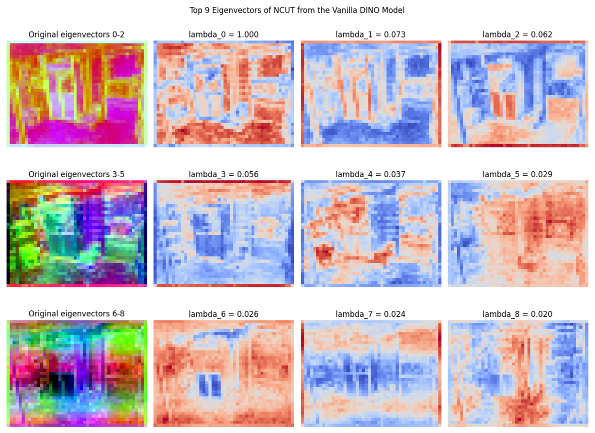

Fig. 7: NCUT Visualizations of DINO Features Originally Produced by the DINO Encoder for the Above Image

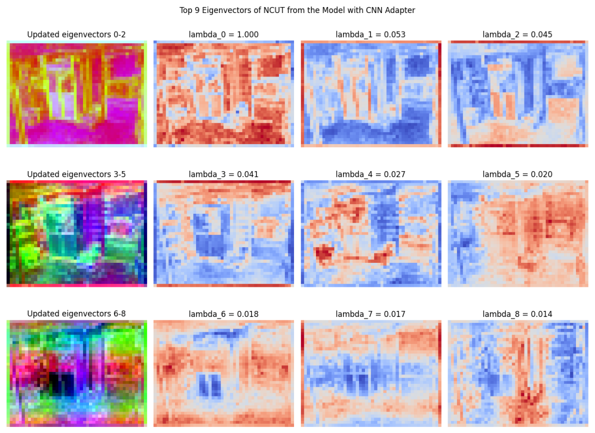

Fig. 8: NCUT Visualizations of DINO Features After Being Modified by our Unregularized CNN Adapter

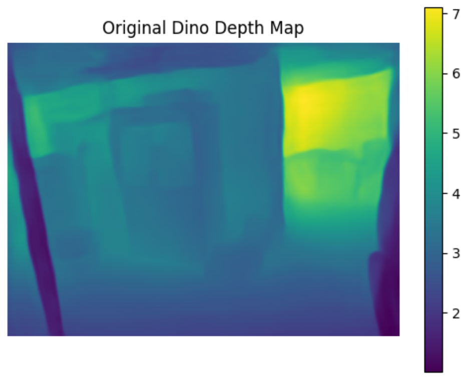

Fig. 9: Original Output Depth Map Produced by Vanilla DINOv2-small, Depth in Meters Indicated by Colorbar

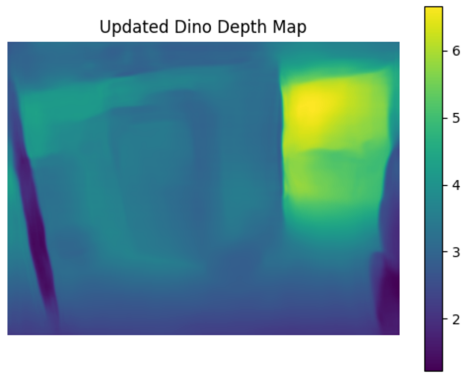

Fig. 10: Updated Output Depth Map Produced with our Unregularized CNN Adapter Applied to DINOv2-small, Depth in Meters Indicated by Colorbar

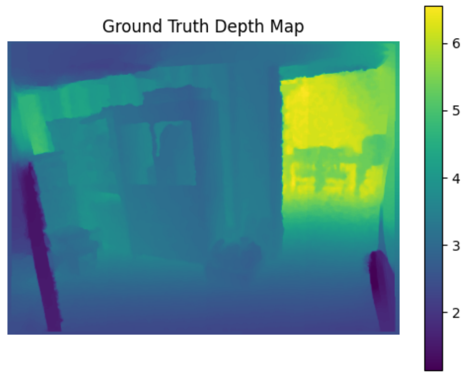

Fig. 11: Ground Truth Depth Map, Depth in Meters Indicated by Colorbar

D. NCUT Visualizations - Regularized CNN Adapter

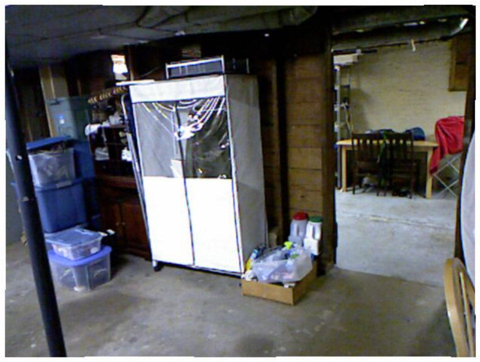

Fig. 12: Input RGB Image from NYUv2 Depth Dataset

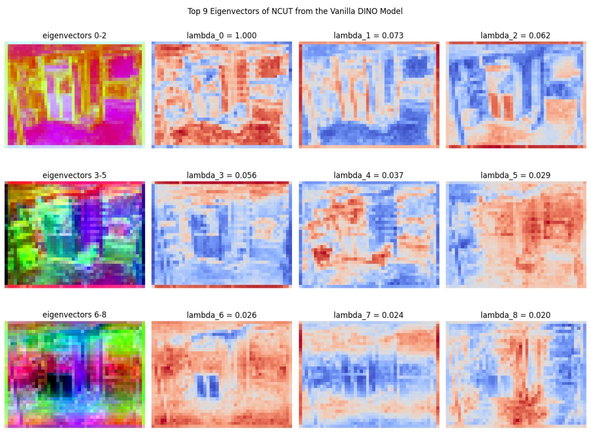

Fig. 13: NCUT Visualizations of DINO Features Originally Produced by the DINO Encoder for the Above Image

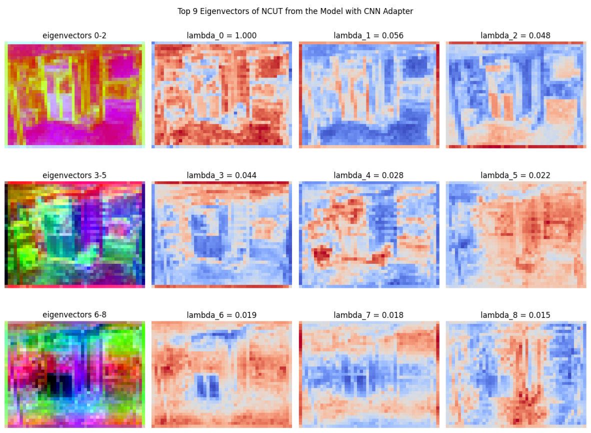

Fig. 14: NCUT Visualizations of DINO Features After Being Modified by our Regularized CNN Adapter

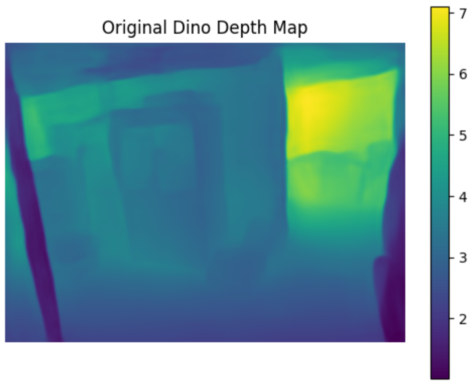

Fig. 15: Original Output Depth Map Produced by Vanilla DINOv2-small, Depth in Meters Indicated by Colorbar

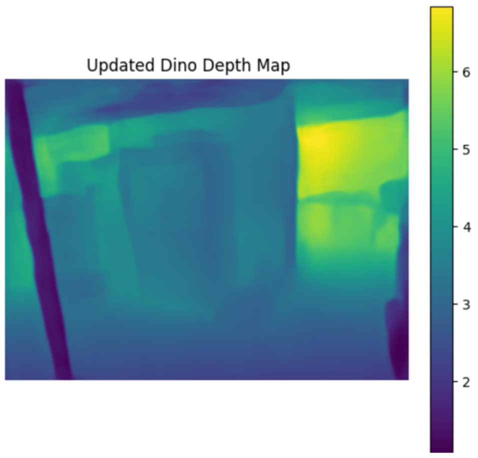

Fig. 16: Updated Output Depth Map Produced with our Regularized CNN Adapter Applied to DINOv2-small, Depth in Meters Indicated by Colorbar

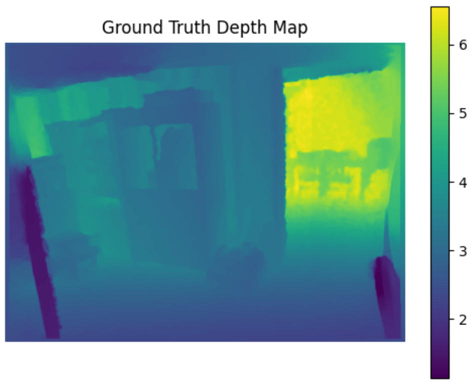

Fig. 17: Ground Truth Depth Map, Depth in Meters Indicated by Colorbar