# Test-time Domain Adaptation for Monocular Depth Estimation

Zhi Li[1,2], Shaoshuai Shi[1], Bernt Schiele[1], Dengxin Dai[1]

***Abstract*— Test-time domain adaptation, i.e. adapting source-pretrained models to the test data on-the-fly in a source-free, unsupervised manner, is a highly practical yet very challenging task. Due to the domain gap between source and target data, inference quality on the target domain can drop drastically especially in terms of absolute scale of depth. In addition, unsupervised adaptation can degrade the model performance due to inaccurate pseudo labels. Furthermore, the model can suffer from catastrophic forgetting when errors are accumulated over time. We propose a test-time domain adaptation framework for monocular depth estimation which achieves both stability and adaptation performance by benefiting from both self-training of the supervised branch and pseudo labels from self-supervised branch, and is able to tackle the above problems: our scale alignment scheme aligns the input features between source and target data, correcting the absolute scale inference on the target domain; with pseudo label consistency check, we select confident pixels thus improve pseudo label quality; regularisation and self-training schemes are applied to help avoid catastrophic forgetting. Without requirement of further supervisions on the target domain, our method adapts the source-trained models to the test data with significant improvements over the direct inference results, providing scale-aware depth map outputs that outperform the state-of-the-arts. Code is available at https://github.com/Malefikus/ada-depth.**

## I. INTRODUCTION

Continuous depth estimation from monocular videos is one of the many fundamental tasks for the development of navigation systems, and is particularly important in autonomous driving scenarios. The development of deep learning enables the acquisition of neural networks learned from large training corpus which can then be used to make inference on the test data. In real world applications, however, the training data (source) are not always sufficient for the model to perform well enough on the test data (target), especially when the domain gap is large, e.g. when the camera setups change, when locations and weather conditions are changing, etc. To tackle this issue, domain adaptation techniques are introduced [1], [2].

Specifically, to acquire better inference from the source-trained model, one needs to further fine-tune the network on the test data in an unsupervised or self-supervised manner, and more practically in an online fashion, i.e. making prediction on the current frame directly after the adaptation to this frame, then moving on to the next frame. In addition, due to privacy concerns, legal constraints or technical reasons, the source data are generally considered unavailable during the adaptation process, making it more challenging but expanding its applicability. In general, source-free test-time domain adaptation is crucial and practical to real world machine perception applications.

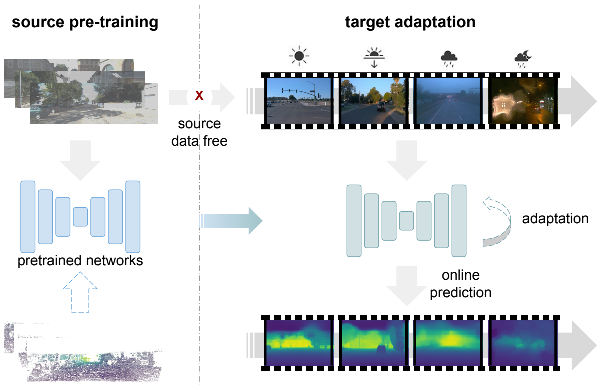

Fig. 1. Overview of our test-time domain adaptation framework. We adapt our source-trained network to the changing target data during test time in an online fashion, without requiring the access of the source data anymore.

We tackle the problem of domain adaptation for depth estimation under this challenging source-free, online setup with continuously changing environment. Fig. 1 is a brief recap of our framework. On the source domain, we train networks on the training videos and the corresponding Lidar groundtruth depth maps. To adapt the source-trained networks to the target domain, we fine-tune the source-pretrained networks with the aligned test images consecutively in a temporal order on-the-fly. Our framework is designed for stable, long-term adaptation without catastrophic forgetting but still with a lot of accuracy improvement. Note that our method acquires scale-awareness through distilling knowledge from scale generalisable source-trained models; therefore, unlike test-time domain adaptation works such as [1], our method does not require extra scale supervision (e.g. velocity supervision) from the test set during adaptation, nor access to the source data, which expands its applicability on low-cost systems. In addition, unlike [1] which requires separately adapting to short sequences, our method continuously adapts to long sequences without degradation in performance.

In summary, our contributions are as follows:

- We develop a test-time domain adaptation framework for continuous depth estimation from monocular videos that significantly improves the inference quality of source-trained models on target data and outperforms the state-of-the-arts. Unlike previous works, our framework is able to produce scale-aware depth predictions on the target data without requiring additional supervision from either target domain or source domain, thanks to our novel 3-branch network taking advantage of both

[1] Max Planck Institute for Informatics
[2] Saarland University Campus

supervised and self-supervised models.

- We propose a simple yet effective pixel scale alignment scheme between source and target data based on geometrical constraints, which significantly improves the source-model inference quality on the target domain already before the adaptation.
- A novel consistency checking technique is proposed to filter out erroneous pixels in the pseudo labels during adaptation, which improves the pseudo label quality thus enhances the adaptation performance.
- We enable long-term adaptation without catastrophic forgetting by proposing an effective regularisation scheme integrated into the effective EMA self-training scheme in the adaptation process.

## II. Related Works

### A. Domain Adaptation

Domain adaptation, or more specifically referred to as unsupervised domain adaptation (UDA), aims to improve the model trained on (usually labelled) source domain for the inference on the unlabelled target domain [3], [4]. There are various techniques for adaptation, such as input alignment [5], [6] and feature distribution alignment between two domains [7], [8], [9]. In addition to feature alignment, there are self-training techniques [10], [11], [12], [13] which fine-tune the model to the target domain using pseudo labels created by the model itself. Under different additional constraints, there are different variants of domain adaptation tasks.

*1) Test-Time Domain Adaptation:* test-time domain adaptation adds a constraint to the general domain adaptation by restricting the access to the source domain data during adaptation [14], [15]. but usually assumes an offline scenario where all the test data are provided for network fine-tuning, making the applicability limited.

*2) Continuous Domain Adaptation:* continuous domain adaptation does not limit the target domain to a specific one, but assumes continually changing target data [2]. Most of the works, however, require access to data from both the source and target domains in order to align the distributions. [2] tackles the problem of source-free, continuous test-time domain adaptation, but only for classification task. For the specific task of depth estimation under this setup, there is not a lot research done, except for [1] which still requires additional supervision (ground truth velocity) from the target data in order to generate correctly scaled depth maps.

### B. Monocular Depth Estimation

Monocular depth estimation aims to predict dense depth maps from single-view RGB images or videos. A lot of research is done in training neural networks by either supervised or self-supervised means.

*1) Supervised Monocular Depth Estimation:* supervised monocular depth estimation methods assume the availability of paired input images and groundtruth Lidar points and utilise the sparse Lidar points as supervisory signals to train models that can produce dense depth maps. A lot of neural network architectures are designed to learn dense depth map from Lidar groundtruth [17], [18], [19], [20], [21], [22], [23]. However, the high cost of Lidar makes it hard to assume the groundtruth to be always available, which is a significant drawback of the supervised methods.

*2) Self-supervised Monocular Depth Estimation:* Self-supervised monocular depth estimation methods generally formulate the depth estimation task as a view synthesis problem [24], [25], where depth and relative camera pose estimators are trained jointly to calculate the view synthesis losses. Improvements are done on top of this basic training framework, for example the utilisation of geometrical consistency through auxiliary optical flow estimations [26], [27], additional weak supervisions such as velocity [28], or temporal cost volume calculation [29]. Although self-supervised methods achieve promising results without requirement of groundtruth Lidar points during training thus can be easily deployed in domain adaptation scenarios, it has a fatal drawback – the output depth maps are always up to an unknown scale, meaning that we still need the groundtruth Lidar points of the test data to apply scale alignment during inference, unless we develop reliable scale recovery techniques (such as in [30]), which is non-trivial.

## III. Method

In this section we show the detailed design of our test-time domain adaptation framework for continuous depth estimation. Given monocular video data with corresponding groundtruth Lidar points in the source domain, we first train two source models using existing supervised and self-supervised depth estimation training scheme, respectively. Note that the supervised model is able to produce absolute scale depth maps, while the self-supervised model does not have any scale awareness, by principle. During adaptation, as is shown in Fig. 2, we first initialise three branches (a regularisation branch, a supervised branch and a self-supervised branch) using the source-trained supervised model (for the first two) and the self-supervised model (for the last). When new data come, the self-supervised branch can be updated in the way it is originally trained, then be used to generate a pseudo label; the regularisation branch produces another pseudo label for regularisation. The two pseudo labels are then compared with the prediction made by the supervised branch, to filter out less confident pixels. The filtered pseudo labels are then used to calculate losses thus update the supervised branch. The updated supervised branch makes final predictions for this frame, and the framework moves on to the next frame.

Our 3-branch adaptation framework is designed in a way that we can benefit from the knowledge distilled from both the supervised and the self-supervised models. Research [31] has shown that disentangling the supervised and self-supervised training conduces the complementary advantages of both loss functions, while training only one model mixing both the losses results in inheriting the limitations from both. For our test-time domain adaptation problem, specifically, the supervised branch provides scale-aware, robust depth predictions which can be further improved by self-training

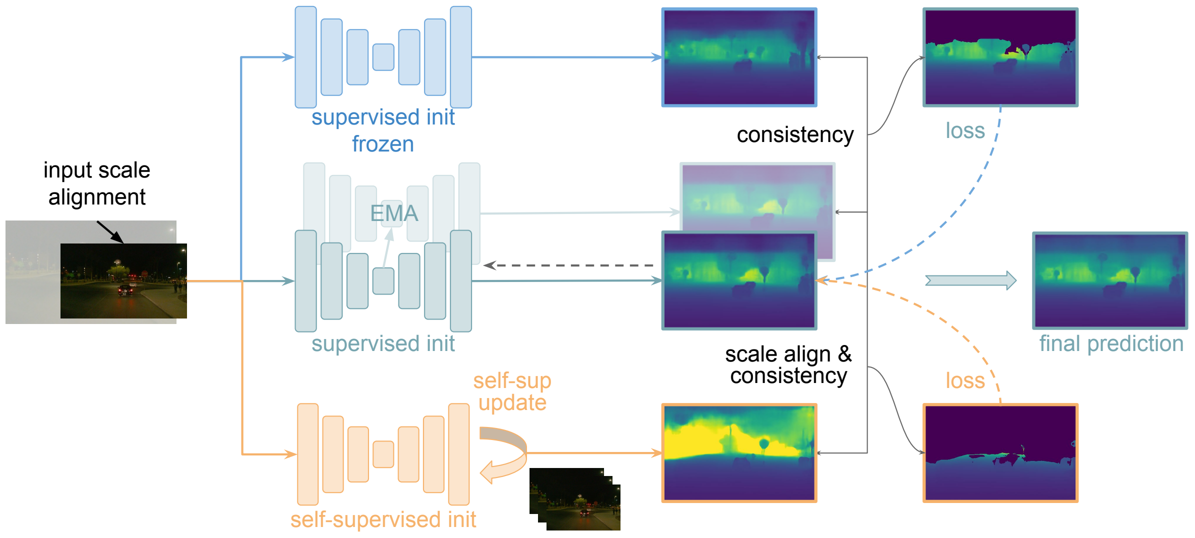

Fig. 2. Pipeline of our adaptation framework. The three branches (from top to bottom) are initialised by source-trained supervised model/ supervised model/ self-supervised model, respectively. For every frame of the test data, the self-supervised branch (bottom) is firstly updated by the unsupervised image synthesis loss which requires only 2 adjacent RGB frames, then be used to create a pseudo label. The regularisation branch (top) generates another pseudo label. The supervised branch (middle) makes a prediction which is then compared with the two pseudo labels, to filter out less confident pixels and create more robust pseudo labels. These pseudo labels are used to update the supervised branch. To increase stability we adopt the EMA [16] self-training scheme for supervised branch. After the iteration, the supervised branch makes an accurate, scale-aware final prediction, and the networks move on to the next frame. Some network details are omitted for simplicity and will be introduced in the texts.

techniques during adaptation, but the improvement is limited; the self-supervised branch generates scale-ambiguous depth estimation, but can be quickly adapted to the test data via self-supervised loss with significant performance improvement. With the two sets of pseudo labels created by the two branches, we improve the accuracy of depth predictions on the test data while retaining the scale-awareness and robustness, even for long-term adaptation.

## A. *Supervised Branch*

For the supervised branch in our framework, we pretrain a single-frame monocular depth estimation network in a supervised way on the source data, following the network architecture in [22]. It consists of a powerful swin-transformer [32] encoder, and a hierarchical decoder comprising four levels of neural window FC-CRF modules [22]. A supervised loss $\mathscr{L}_s$ is defined between the network predictions $\hat{d}$ and groundtruth sparse Lidar points $d^*$. Following common practices in previous works [33], [18], [20], [22], we adopt the Scale-Invariant Logarithmic (SILog) loss proposed by [17]. First, the logarithm difference between $\hat{d}$ and $d^*$ is calculated on each pixel $i$ of the $K$ pixels where the groundtruth Lidar points are available:

$$\Delta d_i = \log \hat{d}_i - \log d_i^*, \tag{1}$$

then the SILog loss is computed as:

$$\mathscr{L}_s = \alpha \sqrt{\frac{1}{K}\sum_i \Delta d_i^2 - \frac{\lambda}{K^2}(\sum_i \Delta d_i)^2}, \tag{2}$$

where $\lambda$ is a variance minimising factor and $\alpha$ is a weight constant controlling the scale of the loss. Following previous works [18], [22], we set $\lambda$ to 0.85 and $\alpha$ to 10.

## B. *Self-Supervised Branch*

We train our self-supervised depth estimation branch on monocular source videos following the Monodepth2 [25] pipeline, where two separate networks – a depth network and a pose network – are jointly trained. For the network architectures we follow [25] and adopt two separate ResNet50 [34] encoders and task specific convolutional decoders for each. The depth network receives an RGB image as input and produces its corresponding depth map; the pose network takes adjacent frames (in our experiments, 2 frames) as input and predicts the relative poses between the frames. The predicted depth map and relative poses can be used to warp one image frame to another; then, a loss function can be defined on the synthesised image and the target image by calculating their photometric differences. For more detailed explanations of the network architecture and losses please refer to [25].

## C. *Target Domain Scale Alignment*

In domain adaptation scenarios, most of the times there are different camera setups between source and target data, resulting in different image sizes. Given that the real world scale does not vary, larger image sizes stand for larger object sizes on the image, which, by intuition, results in different scale estimations for the predicted depth map. We seek to align the pixel scales between the source and target images in order to fix the image/real world proportions, thus avoid drastic disparity between estimated scales on the source and target data. We assume that we do not have access to the source data, but only the metadata (camera intrinsics, camera height to the ground, etc.) of the source and target data.

According to principles of imaging [35], larger focal length results in bigger object size on the image. Therefore, given a target image $I_t(H_t, W_t)$ with height $H_t$ and width $W_t$ whose unit is aligned with the focal length $f_t$, and a known source camera focal length $f_s$, we can align the target image "pixel sizes" (i.e. object sizes) to the source image by simply resizing the image by the proportion of the focal lengths:

$$\hat{I}_t = I_t(H_t \cdot \frac{f_s}{f_t},\ W_t \cdot \frac{f_s}{f_t}). \tag{3}$$

Another important factor which affects the scale of imaging is camera height. In autonomous driving scenes, the imaging of the stretch of the pavement is an important reference to absolute scale estimation; we seek to further align the pixel sizes of the roads on the images between source and target data, to avoid scale drifting during inference. Specifically, for a stretch of road with a certain distance to the camera, the two cameras whose heights are $h_1$ and $h_2$ share the same focal length (or the focal lengths are "aligned" for the images with Equation 3), the sizes of this stretch of road on the images are $i_1$ and $i_2$, respectively. According to principles of triangularity, we can derive that the sizes of the road on the images are proportional to the camera heights:

$$\frac{h_1}{h_2} = \frac{i_1}{i_2}. \tag{4}$$

With Equation 4 we can further align the pixels of $\hat{I}_t$ to the source, given the camera height of the source data $h_s$ and target data $h_t$. The complete scale alignment process will be simply resizing the target images according to the focal lengths and camera heights:

$$\hat{I}'_t = I_t(H_t \cdot \frac{f_s}{f_t} \cdot \frac{h_s}{h_t},\ W_t \cdot \frac{f_s}{f_t} \cdot \frac{h_s}{h_t}). \tag{5}$$

## *D. Continuous Test-Time Adaptation*

After the source models being trained and the target images sizes aligned, we perform our adaptation on the test videos using the pipeline described in Fig. 2. The supervised branch and self-supervised branch are initialised by the supervised and self-supervised models trained on the source data, respectively, and will be updated during adaptation; the regularisation branch is a copy of the source trained supervised model and is kept unchanged during the adaptation process. In every iteration, we first update the self-supervised branch with the self-supervised loss computed on this frame, then make prediction on this frame to be used as a pseudo label; the regularisation branch produces another pseudo label. Both pseudo labels are compared with the prediction of the supervised branch, to filter out less reliable pixels via consistency check. The filtered pseudo labels are then used to update the supervised branch. In addition, an EMA self-training scheme is adopted while updating the supervised branch, to further improve performance gain. The updated supervised branch makes final prediction for this frame, and the adaptation moves on to the next frame. Details of each component are listed below:

*1) Pseudo Label Consistency:* A consistency checking scheme is performed on the pseudo labels created by the self-supervised branch and the regularisation branch, which is conducted by computing the per-pixel difference between one pseudo label $D_p$ (either from the regularisation branch or the self-supervised branch) and the corresponding prediction from the supervised branch $D_s$ and masking out the pixels that have bigger difference. The resulting valid mask $M$ is computed as the Iverson bracket:

$$M = \left[ \frac{\left\| D_s - \gamma D_p \right\|^2}{\gamma D_p} < \sigma \right], \tag{6}$$

where $\sigma$ is the threshold which we empirically set to 0.4 in our experiments, and $\gamma$ is the alignment factor. For regularisation branch, $\gamma$ is set to 1. For the pseudo labels created by self-supervised branch which are scale-ambiguous, we align them to the predictions from supervised branch by median scaling:

$$\gamma = \frac{median(D_s)}{median(D_p)}. \tag{7}$$

Then the pseudo labels are filtered to:

$$\hat{D}_p = \gamma M D_p. \tag{8}$$

*2) EMA Self-Training:* When updating the supervised branch, we adopt the EMA self-training scheme [16] for further performance gain and better robustness. To generate $D_s$, instead of directly using the updated supervised branch, we predict from a slow copy of the supervised model parameters $\theta_t$ at training step $t$ by its exponential moving average over time, defined as:

$$\theta'_t = \alpha \theta'_{t-1} + (1 - \alpha)\theta_t, \tag{9}$$

where $\alpha$ is the smoothness factor which we empirically set to 0.99 in our experiments. For more details of EMA training technique please refer to [16].

*3) Adaptation Loss:* we update the supervised branch by computing SILog loss between its output $D_s$ and the filtered pseudo labels created by the two branches:

$$\mathscr{L}_{ada} = \mathscr{L}_s(D_s, \hat{D}_{ps}) + \mathscr{L}_s(D_s, \hat{D}_{pr}), \tag{10}$$

where $\hat{D}_{ps}$ is the filtered pseudo label created by the supervised branch and $\hat{D}_{pr}$ is the one by the regularisation branch. $\mathscr{L}_s$ is the SILog loss defined in Equation 2.

## IV. EXPERIMENT

We perform extensive experiments and ablation study on various datasets to demonstrate the effectiveness of our framework. This section describes our experimental setup and demonstrates the results and discussions. More technical details of our implementation can be found in our code release.

TABLE I

**RESULTS ON DDAD [28] VALIDATION SET (CROSS-DATASET).** WE REPORT THE ABSOLUTE SCALE RESULTS (WITHOUT MEDIAN SCALING) OF SOTA METHODS, SOTA + OUR SCALE ALIGNMENT SCHEME (INDICATED AS +SA), AND OUR METHOD. BEST IS ILLUSTRATED IN **BOLD**.

| method | target adaptation | source supervision | lower is better | | | | higher is better | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | absRel | sqRel | RMSE | RMSE log | $\sigma < 1.25$ | $\sigma < 1.25^2$ | $\sigma < 1.25^3$ |
| DNet [30] | - | - | 0.383 | 5.297 | 13.757 | 0.518 | 0.385 | 0.675 | 0.824 |
| NewCRF [22] | - | ✓ | 0.437 | 8.211 | 18.498 | 0.817 | 0.249 | 0.362 | 0.480 |
| CoMoDA+sup [1] | ✓ | ✓ | 1.297 | 34.085 | 24.355 | 0.825 | 0.082 | 0.177 | 0.374 |
| CoTTA [2] | ✓ | ✓ | 0.441 | 8.313 | 18.598 | 0.829 | 0.246 | 0.358 | 0.473 |
| DNet [30] + SA | - | - | 0.200 | 1.951 | 8.508 | 0.255 | 0.724 | 0.936 | 0.975 |
| NewCRF [22] + SA | - | ✓ | 0.144 | 1.516 | 7.951 | 0.228 | 0.788 | 0.938 | 0.976 |
| DNet [30] + SA + Ada | ✓ | - | 0.191 | 2.644 | 8.400 | 0.232 | 0.775 | 0.935 | 0.974 |
| CoMoDA+sup + SA [1] | ✓ | ✓ | 1.152 | 29.064 | 18.800 | 0.740 | 0.168 | 0.356 | 0.576 |
| CoTTA + SA [2] | ✓ | ✓ | 0.142 | 1.501 | 7.909 | 0.226 | 0.793 | 0.939 | 0.977 |
| **Ours** | ✓ | ✓ | **0.112** | **1.173** | **6.649** | **0.186** | **0.867** | **0.961** | **0.984** |

TABLE II

**RESULTS ON WAYMO [36] DATASET UNDER DIFFERENT TIME OF DAY/WEATHER CONDITIONS (CROSS-DATASET).** ALL METHODS ARE APPLIED **AFTER** OUR INPUT SCALE ALIGNMENT SCHEME, WITHOUT MEDIAN SCALING. BEST IS ILLUSTRATED IN **BOLD**.

| time/weather condition | method | target adaptation | source supervision | lower is better | | | | higher is better | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | absRel | sqRel | RMSE | RMSE log | $\sigma < 1.25$ | $\sigma < 1.25^2$ | $\sigma < 1.25^3$ |
| clear night (sunny-night-5) | DNet [30] | - | - | 0.398 | 7.469 | 16.422 | 0.586 | 0.351 | 0.597 | 0.752 |
| | NewCRF [22] | - | ✓ | 0.199 | 2.116 | 9.558 | 0.272 | 0.607 | 0.903 | 0.972 |
| | DNet [30] + Ada | ✓ | - | 0.245 | 2.776 | 10.154 | 0.322 | 0.410 | 0.875 | 0.965 |
| | CoMoDA+sup [1] | ✓ | ✓ | 0.577 | 8.834 | 13.277 | 0.483 | 0.301 | 0.537 | 0.783 |
| | CoTTA [2] | ✓ | ✓ | 0.199 | 2.110 | 9.551 | 0.271 | 0.607 | 0.904 | 0.972 |
| | **Ours** | ✓ | ✓ | **0.177** | **1.780** | **8.524** | **0.239** | **0.711** | **0.932** | **0.984** |
| rainy incl. day, dawn and night (rainy-5) | DNet [30] | - | - | 0.456 | 8.468 | 15.002 | 0.843 | 0.123 | 0.568 | 0.729 |
| | NewCRF [22] | - | ✓ | 0.244 | 3.439 | 10.021 | 0.374 | 0.601 | 0.815 | 0.890 |
| | DNet [30] + Ada | ✓ | - | 0.471 | 11.119 | 14.635 | 0.468 | 0.409 | 0.716 | 0.836 |
| | CoMoDA+sup [1] | ✓ | ✓ | 0.669 | 11.494 | 13.203 | 0.521 | 0.253 | 0.519 | 0.757 |
| | CoTTA [2] | ✓ | ✓ | 0.246 | 3.495 | 10.043 | 0.379 | 0.598 | 0.814 | 0.886 |
| | **Ours** | ✓ | ✓ | **0.229** | **2.891** | **9.030** | **0.314** | **0.627** | **0.845** | **0.934** |

## A. Datasets

*1) KITTI dataset [37]:* this dataset is captured in the city of Karlsruhe, Germany, including urban, rural and highway areas. The dataset includes 64 sequences of RGB images at 10fps with projected Lidar points as ground truth depth maps with a maximum range of 80m. Improved (denser) depth maps are provided on the KITTI depth estimation benchmark set [38]. We adopt the same data splitting pattern as introduced in [17], which splits the data into 32 seqs for training ($\sim$40k frames) and 32 seqs for validation ($\sim$4k frames). The camera height of this dataset is 1.65m and focal lengths are on average about 750mm.

*2) DDAD dataset [28]:* this dataset is recorded in several cities in the United States and Japan. It contains monocular videos at 10fps and the corresponding accurate ground-truth depth (across a full 360 degree field of view) generated from high-density LiDARs (up to 200m) mounted on a fleet of self-driving cars. The training set contains 150 scenes with a total of 12650 individual samples, and the validation set contains 50 scenes with a total of 3850 samples. The camera height is around 1.63m on average, and focal lengths around 2100mm.

*3) Waymo dataset [36]:* the more recent Waymo (perception) dataset is recorded across a range of conditions in multiple cities in the US, with large geographic coverage within each city. It comprises of images recorded by multiple high-resolution cameras and sensor readings from multiple high-quality LiDAR scanners (75m) mounted on a fleet of self-driving vehicles. It consists of around 1k videos, each about 200 frames at 10fps, under different weather conditions (sunny, rain) and time of day (day, dawn, night). The camera height is 2.12m, and focal lengths around 2000mm. In our experiments, we define several sub-sequences: **sunny-night-5** for the first 5 sequences of "sunny" night in the validation set; **rainy-5** for all of the 5 rainy scene from train set, including 1 during day, 3 in dawn and 1 at night; **all-6** for 6 sequences from the Waymo training set, including one for each time of day and weather ("day", "dawn" or "night"; "sunny" or "rain", all are the first ones from the category); **sunny-day-5** for the first 5 "sunny" "day" scene of the evaluation set.

## B. Evaluation metrics

We evaluate depth predictions using the standard metrics described in [17], which are computed by comparing the predicted depth and ground truth depth of the pixels where Lidar points are available. The error metrics, i.e. absolute relative difference (absRel), square relative difference (sqRel), root mean squared error (RMSE) and RMSE in logarithm space (RMSE log) are considered the lower the better, and for the 3 accuracy metrics where $\sigma$ – the ratio of prediction and groundtruth – being under certain thresholds, higher is

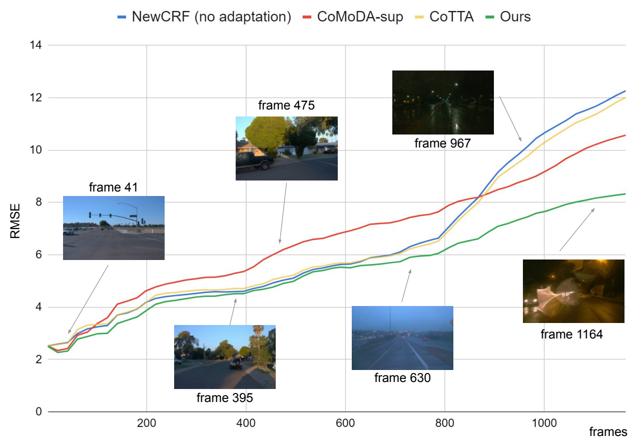

Fig. 3. Accumulated average RMSE (averaged over all previous frames, lower is better) on Waymo "all-6".

TABLE III

Ablative results on Waymo "sunny-day-5" (cross-dataset). We report the results with and without ground truth median scaling ("scale factor" GT and -, respectively). Best is illustrated in **bold**, second best in underline.

| scale factor | method | lower is better | | | |
|---|---|---|---|---|---|
| | | absRel | sqRel | RMSE | RMSE log |
| GT | Monodepth2 (self-sup) | 0.332 | 4.564 | 11.443 | 0.410 |
| | + scale align | 0.229 | 4.697 | 11.314 | 0.277 |
| | + naïve adaptation | 0.209 | 3.631 | 9.345 | 0.249 |
| | NewCRF [22] (sup) | 0.378 | 5.042 | 12.591 | 0.449 |
| | + scale align (focal) | 0.186 | 2.106 | 8.014 | 0.237 |
| | + scale align (height) | 0.157 | 1.615 | 6.853 | 0.210 |
| | + self-sup pseudo | 0.196 | 2.988 | 8.422 | 0.233 |
| | reg pseudo | 0.157 | 1.599 | 6.802 | 0.209 |
| | self-sup+reg pseudo | 0.158 | 1.658 | 6.721 | 0.204 |
| | + pseudo consistency | 0.154 | 1.644 | 6.825 | 0.205 |
| | + ema training | **0.147** | **1.590** | **6.652** | **0.196** |
| - | NewCRF [22] (sup) | 0.482 | 9.234 | 19.413 | 0.858 |
| | + scale align (focal) | 0.291 | 3.424 | 11.408 | 0.403 |
| | + scale align (height) | 0.162 | 2.076 | 7.346 | 0.211 |
| | + self-sup pseudo | 0.188 | 2.550 | 7.917 | 0.227 |
| | reg pseudo | 0.163 | 2.090 | 7.324 | 0.210 |
| | self-sup+reg pseudo | 0.163 | 2.063 | **7.139** | 0.205 |
| | + pseudo consistency | 0.159 | **2.056** | 7.262 | 0.206 |
| | + ema training | **0.155** | 2.130 | 7.241 | **0.202** |

better. For more detailed formulation of these metrics please refer to [17].

In our evaluation, we report both the results with and without the "groundtruth median scaling" operation described in [25], where we scale the predictions with the ratio of the medians of groundtruth and predictions.

### C. Experimental Results

We first evaluate our method on a cross-dataset setting, where we adapt the source models trained on the train set of KITTI Eigen split [17] to the DDAD dataset to get absolute scale predictions, shown in Table I. We observe that directly applying the state-of-the-art methods to the new dataset results in a huge performance drop, making the predictions nearly unreasonable. After applying our simple yet effective input scale alignment scheme, the performance improves on the state-of-the-art methods by a large margin; our method further improves significantly over the enhanced state-of-the-arts, including the naïve adaptation counterpart (noted as "+ Ada") of self-supervised methods. For fair comparison with the same data availability, we report the CoMoDA method [1] with source models trained with the Monodepth2 [25] architecture on the KITTI training set with both supervised and self-supervised loss terms to get scale-aware depth and pose networks; then we apply the CoMoDA adaptation scheme on target data without "velocity supervision" (supervisory on test data) and "replay buffer" (access to source data) to align with our problem setting. We denote this variation as "CoMoDA+sup". Note that our method continuously adapts to the entire DDAD validation set without suffering from catastrophic forgetting like [1].

We further evaluate our method on a more challenging cross-domain setting by applying the KITTI trained source models on the Waymo [36] dataset, shown in Table II. This dataset contains more diverse driving scenes in different cities, different time of day and weather conditions, with quite a different camera setup compared to that of the source data. We evaluate our model on the defined "sunny-night-5" and "rainy-5" sequences. Our method shows significant improvements over existing methods and baselines (all without median scaling, after input scale alignment) in both cases.

Fig. 3 shows how our adaptation method behaves over time on challenging long, changing environments. We show the accumulated average RMSE (RMSE averaged over all previous frames) for the Waymo "all-6" sequences. Our method adapts better to the new changes with less drift.

### D. Ablation Study

We perform ablative study on the Waymo "sunny-day-5" sequences, shown in Table III. We incrementally add every component to the baseline (the supervised branch of our method, based on [22]). The results show that every design choice helps with enhancing the performance, and our full model acquires the best performance.

## V. CONCLUSIONS

We propose a source-free, online test time domain adaptation method for monocular depth estimation. Our input scale alignment scheme significantly improves the network inference in the presence of large domain gaps even before the adaptation, and can be easily integrated into any deep learning based depth estimation framework. Our 3-branch self-training framework shows superiority over the naïve adaptation of either supervised or self-supervised framework. Our effective regularisation operation keeps the learning stable while not degrading the performance gain. Extensive experiments show that our method achieve state-of-the-art results in various datasets, especially in the challenging scenario of large domain gap and continuously changing environment. The practical setting enables our method to be applied to data-scarce, low-cost systems. An interesting future direction can be to reduce the model sizes and/or lighten the training computations in order to deploy it in real-time applications.

## References

[1] Y. Kuznietsov, M. Proesmans, and L. Van Gool, "Comoda: Continuous monocular depth adaptation using past experiences," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2907–2917.

[2] Q. Wang, O. Fink, L. Van Gool, and D. Dai, "Continual test-time domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7201–7211.

[3] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE transactions on neural networks*, vol. 22, no. 2, pp. 199–210, 2010.

[4] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[5] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*. Pmlr, 2018, pp. 1989–1998.

[6] Y. Yang and S. Soatto, "Fda: Fourier domain adaptation for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4085–4095.

[7] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[8] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[9] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7472–7481.

[10] L. Hoyer, D. Dai, and L. Van Gool, "Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9924–9935.

[11] Q. Lian, F. Lv, L. Duan, and B. Gong, "Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: A non-adversarial approach," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6758–6767.

[12] Q. Wang, D. Dai, L. Hoyer, L. Van Gool, and O. Fink, "Domain adaptive semantic segmentation with self-supervised depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8515–8525.

[13] Y. Zou, Z. Yu, B. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 289–305.

[14] J. N. Kundu, N. Venkat, R. V. Babu, *et al.*, "Universal source-free domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4544–4553.

[15] S. Yang, Y. Wang, J. van de Weijer, L. Herranz, and S. Jui, "Generalized source-free domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8978–8987.

[16] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[17] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems*, vol. 27, 2014.

[18] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[19] S. Aich, J. M. U. Vianney, M. A. Islam, and M. K. B. Liu, "Bidirectional attention network for monocular depth estimation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 746–11 752.

[20] S. Lee, J. Lee, B. Kim, E. Yi, and J. Kim, "Patch-wise attention network for monocular depth estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 1873–1881.

[21] C. Shu, Z. Chen, L. Chen, K. Ma, M. Wang, and H. Ren, "Sidert: A real-time pure transformer architecture for single image depth estimation," *arXiv preprint arXiv:2204.13892*, 2022.

[22] W. Yuan, X. Gu, Z. Dai, S. Zhu, and P. Tan, "New crfs: Neural window fully-connected crfs for monocular depth estimation," *arXiv preprint arXiv:2203.01502*, 2022.

[23] Z. Li, X. Wang, X. Liu, and J. Jiang, "Binsformer: Revisiting adaptive bins for monocular depth estimation," *arXiv preprint arXiv:2204.00987*, 2022.

[24] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1851–1858.

[25] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[26] Y. Chen, C. Schmid, and C. Sminchisescu, "Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7063–7072.

[27] X. Luo, J.-B. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent video depth estimation," *ACM Transactions on Graphics (ToG)*, vol. 39, no. 4, pp. 71–1, 2020.

[28] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3d packing for self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2485–2494.

[29] J. Watson, O. Mac Aodha, V. Prisacariu, G. Brostow, and M. Firman, "The temporal opportunist: Self-supervised multi-frame monocular depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1164–1174.

[30] F. Xue, G. Zhuo, Z. Huang, W. Fu, Z. Wu, and M. H. Ang, "Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2330–2337.

[31] J. Baek, G. Kim, and S. Kim, "Semi-supervised learning with mutual distillation for monocular depth estimation," *arXiv preprint arXiv:2203.09737*, 2022.

[32] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[33] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4009–4018.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[36] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9710–9719.

[37] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[38] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *2017 international conference on 3D Vision (3DV)*. IEEE, 2017, pp. 11–20.