DISPBENCH: Benchmarking Disparity Estimation to Synthetic Corruptions

Shashank Agnihotri[*,1] Amaan Ansari[*,1] Annika Dackermann[*,1] Fabian Rösch[*,1]
Margret Keuper[1,2]
[1]Data and Web Science Group, University of Mannheim, Germany
[2]Max-Planck-Institute for Informatics, Saarland Informatics Campus, Germany
shashank.agnihotri@uni-mannheim.de

Abstract

Deep learning (DL) has surpassed human performance on standard benchmarks, driving its widespread adoption in computer vision tasks. One such task is disparity estimation, estimating the disparity between matching pixels in stereo image pairs, which is crucial for safety-critical applications like medical surgeries and autonomous navigation. However, DL-based disparity estimation methods are highly susceptible to distribution shifts and adversarial attacks, raising concerns about their reliability and generalization. Despite these concerns, a standardized benchmark for evaluating the robustness of disparity estimation methods remains absent, hindering progress in the field.

To address this gap, we introduce DISPBENCH, *a comprehensive benchmarking tool for systematically assessing the reliability of disparity estimation methods.* DISPBENCH *evaluates robustness against synthetic image corruptions such as adversarial attacks and out-of-distribution shifts caused by 2D Common Corruptions across multiple datasets and diverse corruption scenarios. We conduct the most extensive performance and robustness analysis of disparity estimation methods to date, uncovering key correlations between accuracy, reliability, and generalization. Open-source code for* DISPBENCH.

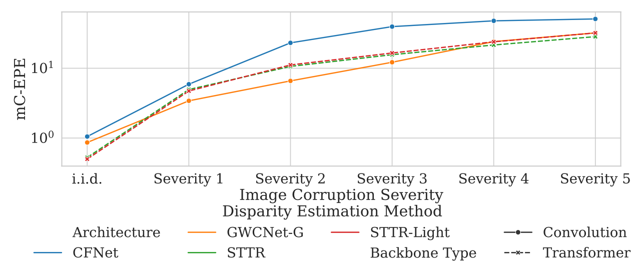

Figure 1. Analyzing the generalization ability of some Disparity estimation methods: GWCNet [26], CFNet [55], and STTR and STTR-light [39] proposed over time. The y-axis represents the mean End-Point-Error (EPE) on Syntheticc Corruptions (2D Common Corrruptions [31]) at different severalties (severity=0 is i.i.d. performance) using the FlyingThings3D [41], i.e., lower is better. We observe that disparity estimation methods lack the generalization ability to common corruptions and, thus, are not safe for real-world deployment.

1. Background

The vision task of disparity estimation, also commonly known as stereo-matching is used to estimate the disparity between matching pixels in stereo image pairs. Mayer et al. [41] proposed the first Deep Learning (DL) based method for disparity estimation called DispNet. This led to disparity estimation becoming primarily a DL-based task [2, 26, 39, 55]. However, DL-based methods are known to be unreliable [1, 3, 13, 20, 47], they tend to learn shortcuts rather than meaningful feature representations [21] and can be easily deteriorated even by small perturbations, causing the evaluation samples to not be independent and identically distributed (i.i.d.) w.r.t. the training samples. This shift from i.i.d. samples can be caused due to changes in the environment, changes in weather conditions, or image corruption due to sensor noise [5, 14, 25, 31, 56, 57, 61]. Such shifts cause the evaluations to be Out-Of-Distribution (OOD), and robustness to such shifts is called OOD Robustness. OOD Robustness is often used as a metric for the generalization ability of a method [23, 24, 32, 33, 37]. Another possible cause of distribution shifts could be either accidental or malicious adversarial attacks [38, 42, 43, 51–53]. Here, the perturbations made to an image are optimized to fool the method while the semantic meaning of the images remains the same for a human observer. When adversarial attacks are optimized with full information about a model and its loss, they are called white-box adversarial attacks. Since these white-box attacks can potentially simulate the worst-case scenario for a method, they are often

Accepted at CVPR 2025 Workshop on Synthetic Data for Computer Vision. *Equal Contribution.

used as a proxy to measuring their reliability [4, 22, 35].

In Fig. 1, we provide an overview of the i.i.d. performance, generalization ability, and reliability of disparity estimation methods proposed over time on the FlyingThings3D dataset [41]. We include old popular methods such as GWCNet and CFNet and new large transformer-based STTR and its lightweight version STTR-light, which, due to its training regime, are proposed as zero-shot disparity estimation methods. Here, we observe a disturbing pattern: while the i.i.d. performance has improved over time, since this improvement has been the focus of most works, the models still lack robustness. This is particularly concerning as disparity estimation is often used in the real world, especially for safety-critical scenarios such as medical surgery procedures [48, 60], including invasive surgeries such as laparoscopy [46] and in autonomous driving [10]. Here, safety is paramount, and to ensure the safe deployment of recent DL-based disparity estimation methods, their reliability and generalization ability need to be guaranteed. However, no such guarantees can be provided currently since no works focus on OOD and the adversarial robustness of disparity estimation methods. This is primarily due to a lack of datasets that enable such studies. Capturing corruptions in the wild and then annotating for disparity estimation is a time and resource intensive process.

Some prior works have focused on other kinds of robustness; for example, a recent work [62] looks into the robustness of disparity estimation works to domain shifts, while [40, 63] studies the robustness of methods to occlusions. Currently, there exists no unified framework to evaluate disparity estimation methods for safe deployment in the real world. Guo et al. [27] recently proposed a benchmarking tool for disparity estimation methods. However, this tool is limited to i.i.d. performance evaluations. This is a significant limitation impeding the community's ability to ensure safe, reliable, and generalizable DL-based disparity estimation methods for the real world.

To bridge this gap, we propose DispBench, the first robustness benchmarking tool for disparity estimation. DispBench is easy to use and extending it to future disparity estimation methods and datasets, when they are proposed, is straightforward. It is inspired by similar popular benchmarks for the image classification tasks [12, 58] and object detection [8, 15, 16, 30, 45]. It enables i.i.d. evaluations of various DL-based disparity estimation methods across multiple commonly used disparity estimation datasets. It also facilitates research in the reliability and generalization ability of disparity estimation methods, as it enables users to use synthetic image corruptions, specifically, 5 diverse adversarial attacks and 15 established common corruptions. This will help researchers build better models that are not limited to improved performance on identical and independently distributed (i.i.d.) samples and are less vulnerable to adversarial attacks while generalizing better to image corruptions. Our proposed DispBench facilitates this, streamlining it for future research to utilize.

The main contributions of this work are as follows:

- We provide a benchmarking tool DispBench to evaluate the performance of most DL-based disparity estimation methods over 2 different datasets and synthetic corruptions.
- We benchmark the aforementioned models against commonly used adversarial attacks and common corruptions that can be easily queried using DispBench.
- We perform an in-depth analysis using DispBench and present interesting findings showing methods that perform well on i.i.d. are remarkably less reliable and generalize worse than other non-well-performing methods.
- We show that synthetic corruptions on synthetic datasets do not represent real-world corruptions; thus, synthetic corruptions on real-world datasets are required.

2. DispBench Usage

There exists no standardized tool for evaluating the performance of disparity estimation methods. Thus, the codebase for such a tool had to be written from scratch. In the following, we describe the benchmarking tool, DispBench. Currently, it supports 4 unique architectures (new architectures to be added to DispBench with time) and 2 distinct datasets, namely FlyingThings3D [41] and KITTI2015 [44] (please refer Sec. 2.4 for additional details on the datasets). It enables training and evaluations on all aforementioned datasets, including evaluations using SotA adversarial attacks such as CosPGD [4] and other commonly used adversarial attacks like BIM [36], PGD [35], FGSM [22], under various Lipshitz (l_p) norm bounds and APGD [59] under the ℓ_∞-norm bound. Additionally, it enables evaluations for Out-of-Distribution (OOD) robustness by corrupting the inference samples using 2D Common Corruptions [31].

We follow the nomenclature set by RobustBench [12] and use "threat_model" to define the kind of evaluation to be performed. When "threat_model" is defined to be "None", the evaluation is performed on unperturbed and unaltered images, if the "threat_model" is defined to be an adversarial attack, for example "PGD", "CosPGD" or "BIM", then DispBench performs an adversarial attack using the user-defined parameters. Whereas, if "threat_model" is defined to be "2DCommonCorruptions", the DispBench performs evaluations after perturbing the images with 2D Common Corruptions. If the queried evaluation already exists in the benchmark provided by this work, then DispBench simply retrieves the evaluations, thus saving computation. Please refer to Appendix C for details on usage.

Following, we show the basic commands to use DispBench. We describe each attack and common corruption supported by DispBench in detail in Appendix C. Please refer to Appendix E for details on the arguments.

2.1. Model Zoo

It is challenging to find all checkpoints, whereas training them is time and compute-exhaustive. Thus, we gather available model checkpoints made available online by the respective authors. The trained checkpoints for all models available in DISPBENCH can be obtained using the following lines of code:

```
from dispbench.evals import load_model
model = load_model(model_name='STTR',
    dataset='KITTI2015')
```

Each model checkpoint can be retrieved with the pair of 'model_name', the name of the model, and 'dataset', the dataset for which the checkpoint was last fine-tuned.

2.2. Adversarial Attacks

To evaluate a model for a given dataset on an attack, the following lines of code are required.

```
from dispcbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015'
 ↪  retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model, for example, when the threat model is a PGD attack, 'config.yml' could contain 'threat_model=*"PGD"*', 'iterations=*20*', 'alpha=*0.01*', 'epsilon=*8*', and 'lp_norm=*"Linf"*'. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False' then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the adversarial attack to be used, given as a string.
  - 'iterations' are the number of attack iterations, given as an integer.
  - 'epsilon' is the permissible perturbation budget ϵ given a floating point (float).
  - 'alpha' is the step size of the attack, α, given as a floating point (float).
  - 'lp_norm' is the Lipschitz continuity norm (l_p-norm) to be used for bounding the perturbation, possible options are 'Linf' and 'L2' given as a string.
  - 'target' is false by default, but to do targeted attacks, either the user can set 'target'=True, to use the default target of $\overrightarrow{0}$, or can pass a specific tensor to be used as the target.

The adversarial attacks supported by DISPBENCH are *FGSM, BIM, PGD, APGD, and CosPGD*.

In Fig. 2, we show example images perturbed using different adversarial attacks and the change in disparity estimation performed by STTR. Here, all attacks are optimized for 20 attack iterations, with α=0.01 and $\epsilon = \frac{8}{255}$ under the ℓ_∞-norm bound.

2.3. 2D Common Corruptions

To evaluate a model for a given dataset with 2D Common Corruptions, the following lines of code are required.

```
from dispbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015',
 ↪  retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model; for example, when the threat model is 2D Common Corruption, 'config.yml' could contain 'threat_model=*"2DCommonCorruption"*', and 'severity=*3*'. Please note, when the 'threat_model' is the common corruption, DISPBENCH performs evaluations on all corruptions under the respective 'threat_model' and returns the method's performance on each corruption at the requested severity. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False' then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the common corruption to be used, given as a string, i.e. '2DCommonCorruption'.
  - 'severity' is the severity of the corruption, given as an integer between 1 and 5 (both inclusive).

DISPBENCH supports the following 2D Common Corruption: 'gaussian_noise', shot_noise', 'impulse_noise', 'defocus_blur', 'frosted_glass_blur', 'motion_blur', 'zoom_blur',

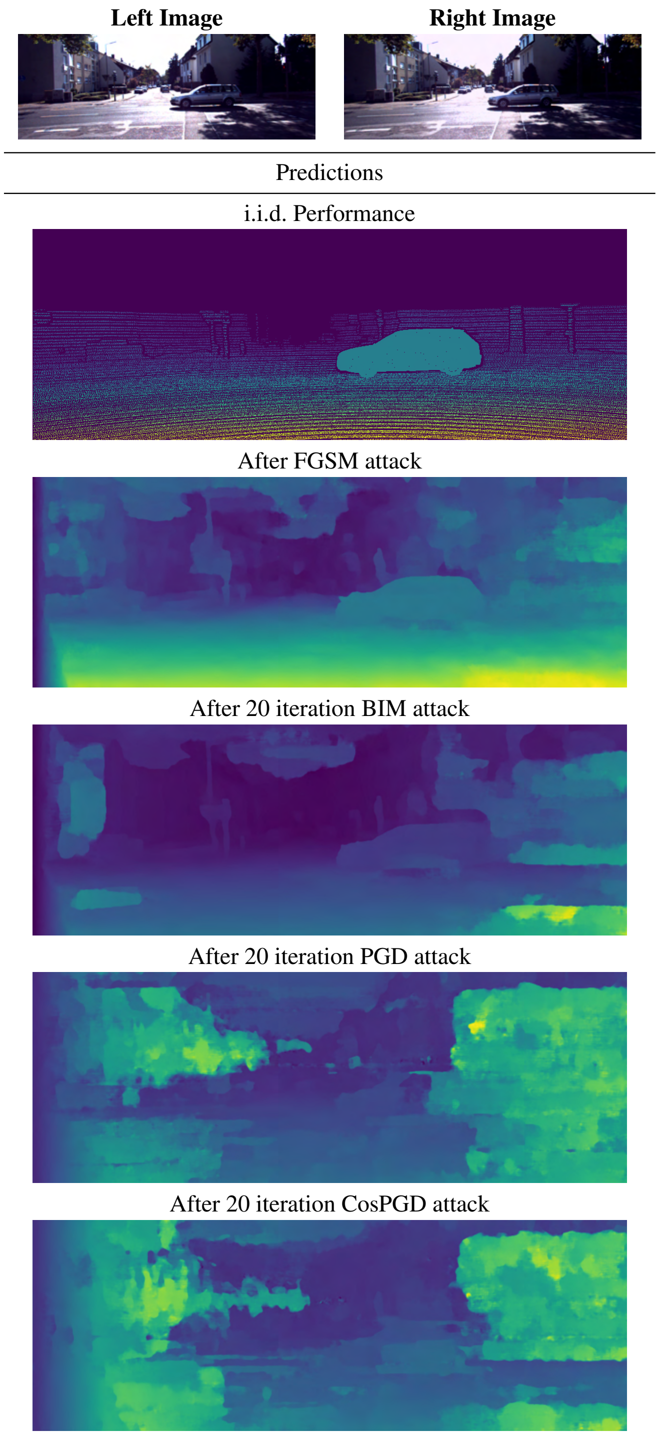

Figure 2. Example of performing adversarial attacks on STTR using KITTI2015 dataset under different attacks. We show the samples before and after the attacks and the predictions before and after the respective adversarial attacks.

'snow', 'frost', 'fog', 'brightness', 'contrast', 'elastic', 'pixelate', 'jpeg'. For the evaluation, DispBench will evaluate the model on the validation images from the respective dataset corrupted using each of the aforementioned corruptions for the given severity and then report the mean performance over all of them.

In Fig. 3, we show example images perturbed using the 2D Common Corruption: Frost and the change in disparity estimation performed by STTR over different severity strengths.

2.4. Dataset Details

DispBench currently supports two distinct disparity datasets. Following, we describe these datasets in detail.

2.4.1. FlyingThings3D

This is a synthetic dataset proposed by [41] largely used for training and evaluation of disparity estimation methods. This dataset consists of 25000 stereo frames, of everyday objects such as chairs, tables, cars, etc. flying around in 3D trajectories. The idea behind this dataset is to have a large volume of trajectories and random movements rather than focus on a real-world application. In their work, [17] showed models trained on FlyingThings3D can generalize to a certain extent to other datasets.

2.4.2. KITTI2015

Proposed by [44], this dataset is focused on the real-world driving scenario. It contains a total of 400 pairs of image frames, split equally for training and testing. The image frames were captured in the wild while driving around on the streets of various cities. The ground-truth labels were obtained by an automated process.

3. Initial Evaluations using DispBench

We use DispBench to perform some initial benchmarking and make some interesting observations. Following, we discuss the details of the benchmarking process. Please note, we use the FlyingThing3D and the KITTI2015 dataset for the benchmarking. However, very few pretrained architectures are available for KITTI2015, and thus our evaluations using KITTI2015 are limited to these. While DispBench enables the training of architectures of multiple datasets, doing so is beyond our resource capabilities.

For additional details on the datasets, please refer to Sec. 2.4.

Measuring Generalization Ability. Inspired by multiple works [12, 32, 33] that use OOD Robustness of methods for evaluating the generalization ability of the method, even evaluate over every common corruption, that is the 15 2D Common Corruptions: 'Gaussian Noise', Shot Noise', 'Impulse Noise', 'Defocus Blur', 'Frosted Glass Blur', 'Motion Blur', 'Zoom Blur', 'Snow', 'Frost', 'Fog', 'Brightness',

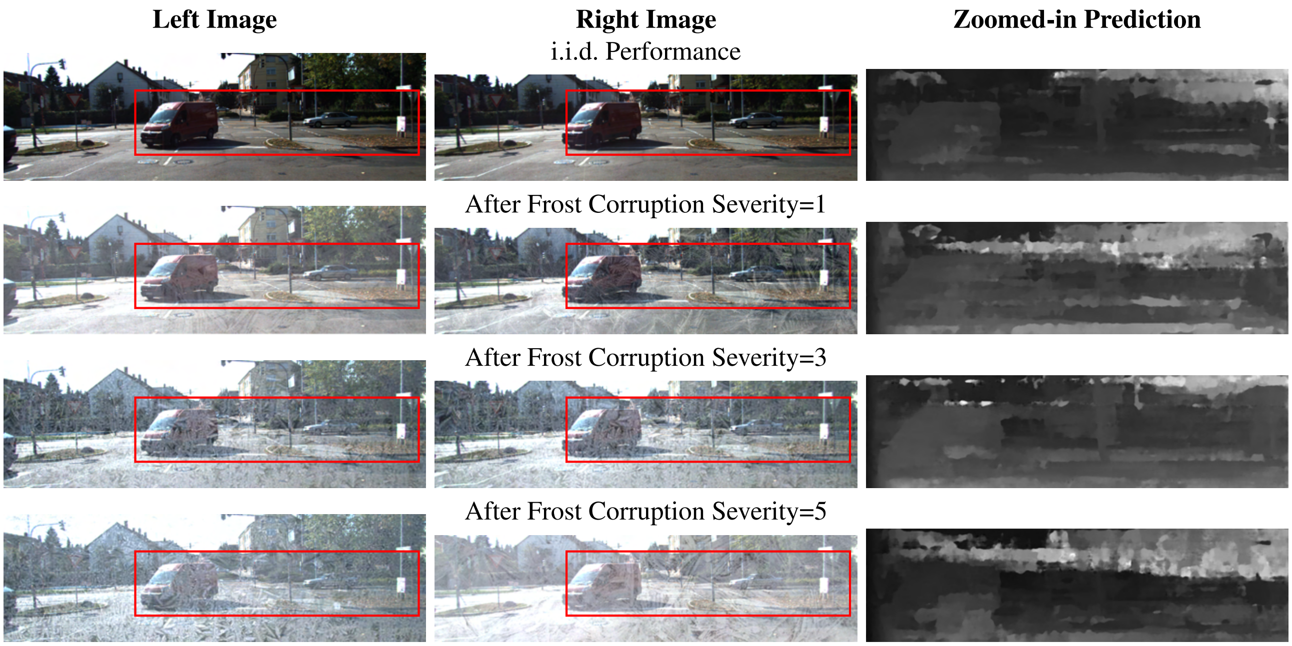

Figure 3. Example of predictions using STTR on KITTI2015 dataset under different severities of the 2D Common Corruption: Frost.

'Contrast', 'Elastic Transform', 'Pixelate', 'JPEG Compression'. Then, we find the mean EPE w.r.t. the ground truth for a given method, across all corruptions at a given severity and report use this to measure the Generalization Ability. We corrupt the pair of stereo images with the same corruption at the same severity when evaluating.

Ideally, one would like to evaluate the generalization ability and reliability of methods using real-world samples captured in the wild. However, annotation of these samples is a challenging and time-consuming task, and thus, no such dataset is available for disparity estimation. Sakaridis et al. [50] captured such data in the wild with domain shifts due to changes in time of day and changes in weather conditions like snowfall, rain, and fog. They also provide pixel-level annotations for their images, however, these annotations are only available for semantic segmentation, and these images are monocular and not stereo. They propose this as the Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding (ACDC) dataset. Interestingly, in their work, Agnihotri et al. [7] showed a very strong positive correlation between the performance of most methods on the ACDC dataset and their performance against in-domain images corrupted with the 2D Common Corruptions to cause a synthetic domain shift. This is an important finding as it proves that 2D Common Corruptions can be used as a proxy to real-world domain shifts. We discuss this in Appendix A. For details on the dataset, please refer to the appendix.

Measuring Reliability Under Adversarial Attacks. Adversarial attacks, especially white-box attacks, serve as a proxy to the worst-case scenario and help understand the quality of the representations learned by a model [4, 53, 59]. DispBench provides the ability to evaluate the models against some popular adversarial attacks, as discussed in Sec. 2.2. However, we focus this work towards realistic corruptions possible in the real world. For evaluations over adversarial attacks, please refer to Appendix F.

Architectures Used. Disparity estimation networks essentially estimate optimal correspondence matching between pixels on epipolar lines in the left and right images to infer depth. Most disparity estimation architectures used a cost volume with cross-correlation or contamination of feature representations for the left and right images. However, **GWCNet-G** [26] proposed using group-wise correlations to construct the cost volume. This leads to a significant boost in i.i.d. performance and inference speed. **CFNet** [55] proposed fusing on multiple low-resolution dense cost volumes to enlarge the receptive field, enabling extraction of robust structural representations, followed by cascading the cost volume representations to alleviate the unbalanced disparity estimation. It was proposed to be robust to large domain differences and was SotA when proposed. **Stereo-Transformers (STTR)**, Li et al. [39] proposes to replace the cost volume construction with dense pixel matching using position information and attention to enable sequence-to-sequence matching. This relaxes the limitation of a fixed

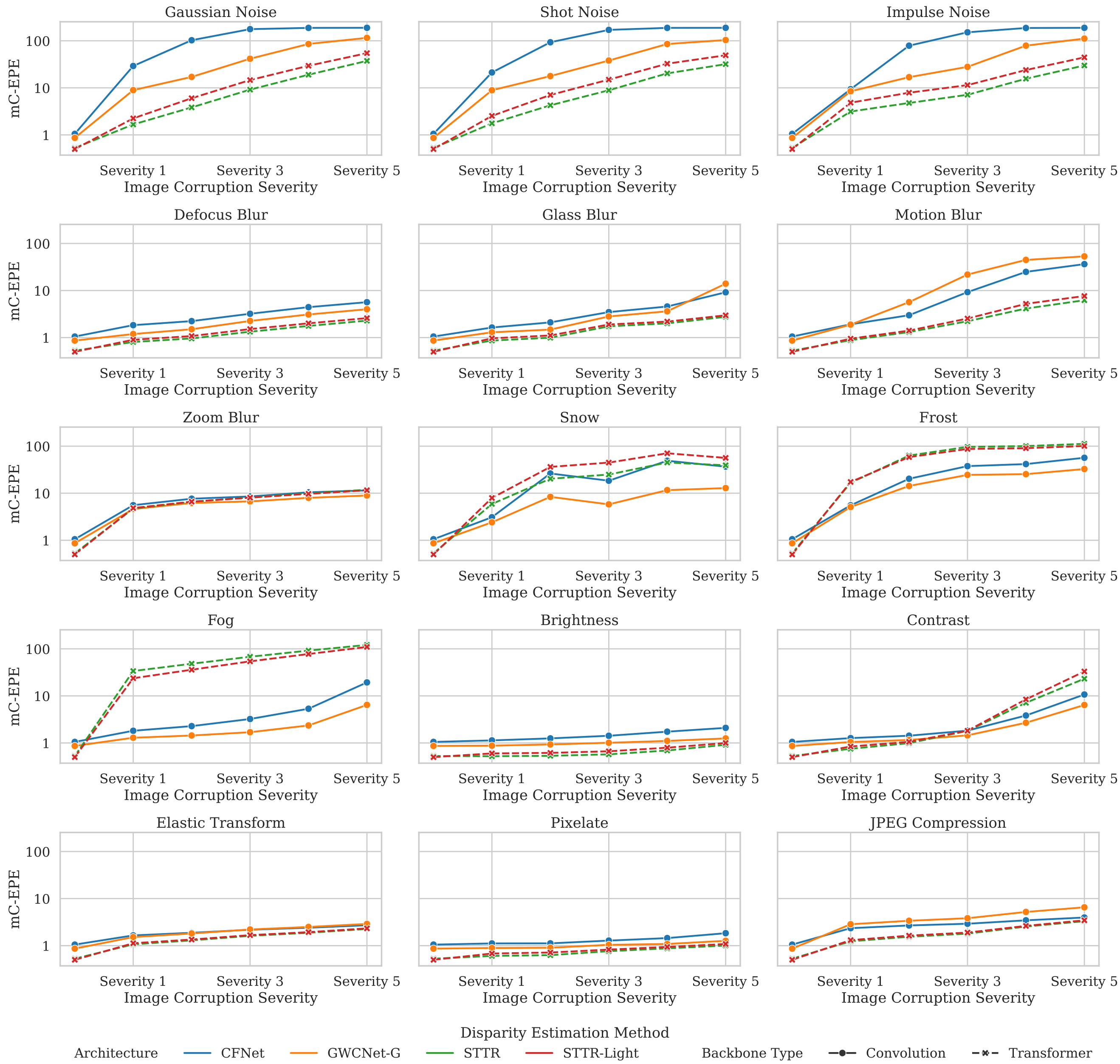

Figure 4. Using the FlyingThings3D dataset for disparity estimation, we perform an initial benchmarking of i.i.d. performance and generalization abilities of four popular disparity estimation methods. CFNet and GWCNet are traditional CNN-based stereo matching methods, whereas STTR and STTR-light are newly proposed transformer-based large models capable of zero-shot disparity estimation. Here, we use their fine-tuned versions for the FlyingThings3D dataset. The y-axis reports the mean EPE over the entire validation set for the respective corruption, and the x-axis denotes the severity of the 2D Common Corruption used to corrupt the input images. We report the i.i.d. performance at severity=0. Here we observe that while all four methods are highly vulnerable to Noise and Weather corruptions, newly proposed STTR and STTR-light are surprisingly less robust than the older CNN-based methods against weather corruptions. This finding is interesting and concerning as weather corruptions are the most likely real-world domain shift.

disparity range and identifies occluded regions with confidence estimates. STTR generalizes across different domains, even without fine-tuning. However, in our evaluations, we use fine-tuned checkpoints for a fair comparison of reliability and generalization capabilities. **STTR-light** is the lightweight version of STTR proposed for faster inference with only a marginal drop in i.i.d. performance. We evaluate using publicly available pre-trained checkpoints.

4. Key Findings

In the following, we present the key findings made using the initial benchmarking using the DispBench.

4.1. FlyingThings3D

Following, we discuss the observations made in the robustness benchmark created using DispBench. We report the evaluations in Fig. 4. Here, we observe that indeed the i.i.d. performance of the new methods like STTR and STTR-light is better than the older CNN-based CFNet and GWCNet-G, however, the same is not always true for their generalization abilities. All four considered methods appear to be robust to digital corruptions such as changes in brightness, contrast, elastic transform, pixelated, and JPEG compression to a significant extent. While, all four methods appear to be extremely non-robust to additive noise, possible in the real world due to sensor error, causing the mean errors to go as high as 100. Please note that compared to the single-digit EPE values for i.i.d. performance, these errors are significantly high.

The most interesting behavior is seen under different weather corruptions: Snow, Frost, and Fog. Here, all four methods are non-robust, however, the newer transformer-based methods STTR and STTR-light are significantly more non-robust. This is quite alarming, as weather corruptions are the most natural domain shifts possible in the real world, and here, the large models fail significantly worse. Especially under Frost and Fog corruptions, the larger STTR performs worse than its lightweight counterpart, STTR-light. This raises some interesting concerns that warrant further study and deeper analysis.

4.2. KITTI2015

There are very limited pre-trained architectures available on KITTI2015 for the disparity estimation task, namely GWCNet-G and STTR. We perform our analysis using these and report the evaluations in Fig. 5. We observe that the newly proposed STTR is less robust than GWCNet-C across all corruptions and severities. This does not align with the observations made with synthetic corruptions on the synthetic dataset FlyingThings3D. This suggests that further analysis is required.

5. Synthetic Corruptions on Synthetic Dataset vs Synthetic Corruptions on Real World Dataset

Following the findings from Sec. 4.2, we investigate whether the performance of models on synthetic corruptions (2D Common Corruptions) on synthetic dataset (FlyingThings3D) can serve as a proxy to the performance of models on synthetic corruptions (2D Common Corruptions) on real-world data (KITTI2015). We report this analysis in Fig. 6 and observe that synthetic corruptions on synthetic datasets do not represent synthetic corruptions on real-world datasets. As known from [7], synthetic corruptions on real-world datasets represent real-world corruptions. By extension, synthetic corruptions on synthetic datasets do not represent real-world corruptions. This crucial finding eliminates the possibility of using synthetic simulators like CARLA [18], LGSVL (SVL Simulator) [49], AirSim [54], and others for applications in the real world.

6. Conclusion

Evaluating a model's robustness is vital for real-world applications. However, capturing corruptions in the real world is time and resource intensive. Here, synthetic corruptions appear to be an attractive alternative. Thus, we propose DispBench, the first robustness benchmarking tool and a novel benchmark on synthetic corruptions for disparity estimation methods. We discuss the unique features of DispBench in detail and demonstrate that the library is user-friendly, such that adding new methods or performing evaluation is very intuitive. We use DispBench to evaluate the i.i.d. performance and OOD generalization of some popularly used disparity estimation methods. We observe that under realistic scenarios, recently proposed large transformer-based methods known to be SotA on i.i.d. samples do not generalize well to image corruptions, demonstrating the gap in current research when considering real-world applications. Lastly, we show experimentally that synthetic corruptions on synthetic datasets do not represent real-world corruptions, thus, synthetic corruptions on real-world datasets present a more promising path. DispBench enables a deeper understanding of the reliability and generalization abilities of disparity estimation methods, and its consolidated nature facilitates more streamlined research.

Future Work. Very recently, OpenStereo [27] has been made public that supports newly proposed stereo matching methods, which are foundational models for stereo matching like StereoAnything [29], and LightStereo [28]. We intend to adapt our evaluator into OpenStereo to enable safety studies of SotA disparity estimation methods. Additionally, more in-depth analysis of the disparity estimation methods, for example, as done by [19] for classification methods, would help understand the models and their workings, especially in terms of their robustness performance.

Limitations. Benchmarking disparity estimation methods is a compute and labor-intensive endeavor. Thus, best utilizing available resources, we currently use DispBench to benchmark a limited number of settings, using the most popular works for now. The benchmarking tool itself offers significantly more combinations that can be benchmarked.

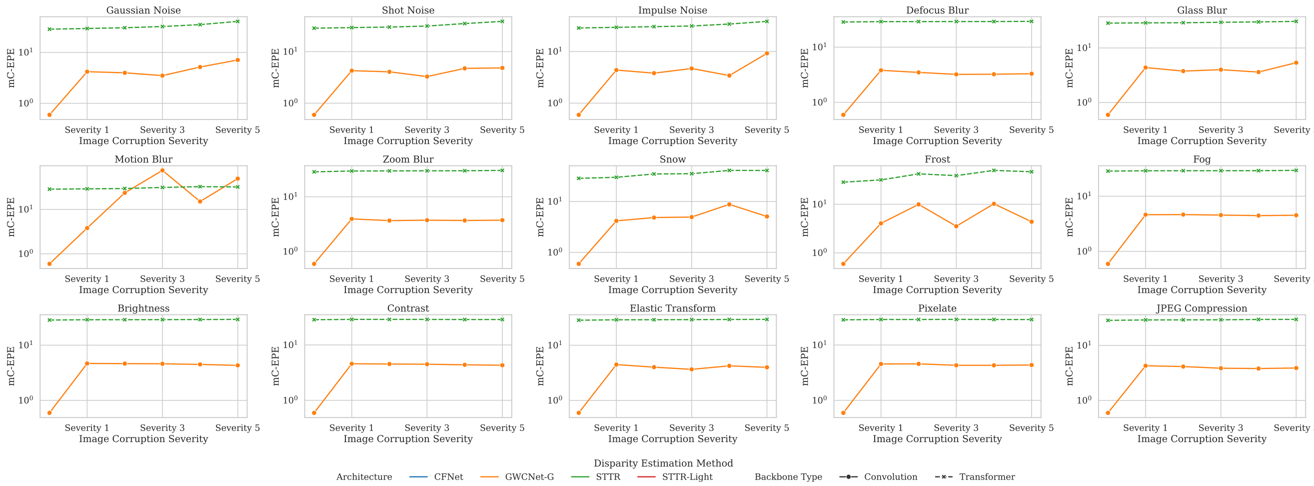

Figure 5. Using the KITTI2015 dataset for disparity estimation, we perform an initial benchmarking of i.i.d. performance and generalization abilities of the two popular and available disparity estimation methods. GWCNet is a traditional CNN-based stereo matching method, whereas STTR is a newly proposed transformer-based large model capable of zero-shot disparity estimation. Here, we use their fine-tuned versions for the KITTI2015 dataset. The y-axis reports the mean EPE over the entire validation set for the respective corruption, and the x-axis denotes the severity of the 2D Common Corruption used to corrupt the input images. We report the i.i.d. performance at severity=0. Here, we observe that while both the methods are highly vulnerable to Noise and Weather corruptions, the newly proposed STTR is surprisingly less robust than the older CNN-based method against all corruptions. This finding is interesting and concerning as it contradicts the findings on the Synthetic Dataset FlyingThings3D in Fig. 4.

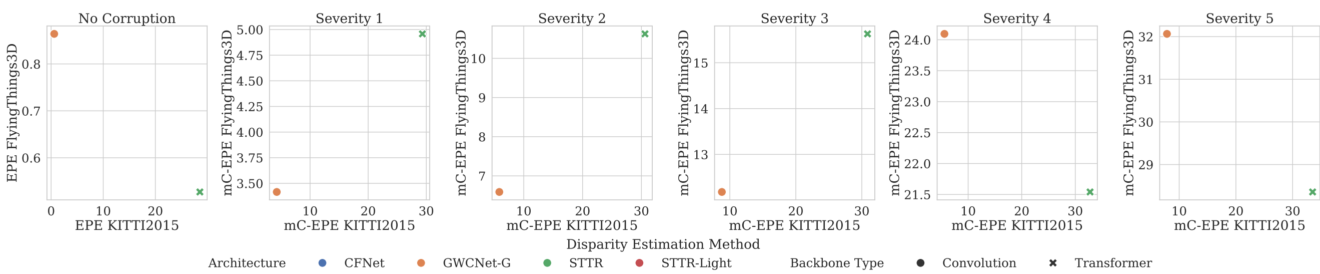

Figure 6. For the same architecture, we report the mean EPE across all corruptions for a checkpoint pretrained on Flyingthings3D against synthetic 2D Common Corruption on Flyingthings3D and correlate its performance with the checkpoint trained on KITTI2015 against synthetic 2D Common Corruptions on KITTI2015. For individual corruptions, please refer to Fig. 9. We observe no correlation in performance, indicating that synthetic corruptions on synthetic datasets cannot be used as proxy for real-world corruptions.

Reproducibility Statement

Every experiment in this work is reproducible and is part of an effort toward open-source work. DISPBENCH will be open-source and publicly available, including all evaluation logs and model checkpoint weights. This work intends to help the research community use synthetic corruptions to build more reliable and generalizable disparity estimation methods such that they are ready for deployment in the real world even under safety-critical applications. The open-source code and model weights for DISPBENCH is available here: https://github.com/shashankskagnihotri/benchmarking_robustness/tree/disparity_estimation/final/disparity_estimation.

Acknowledgments

M.K. acknowledges funding by the DFG Research Unit 5336. The authors gratefully acknowledge support by the state of Baden-Württemberg through bwHPC and computing time provided on the high-performance computer HoreKa by the National High-Performance Computing Center at KIT (NHR@KIT). This center is jointly supported by the Federal Ministry of Education and Research and the Ministry of Science, Research, and the Arts of Baden-Württemberg, as part of the National High-Performance Computing (NHR) joint funding program. HoreKa is partly funded by the German Research Foundation (DFG).

References

[1] Shashank Agnihotri, Kanchana Vaishnavi Gandikota, Julia Grabinski, Paramanand Chandramouli, and Margret Keuper. On the unreasonable vulnerability of transformers for image restoration-and an easy fix. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3707–3717, 2023. 1

[2] Shashank Agnihotri, Julia Grabinski, and Margret Keuper. Improving stability during upsampling – on the importance of spatial context, 2023. 1

[3] Shashank Agnihotri, Julia Grabinski, Janis Keuper, and Margret Keuper. Beware of aliases–signal preservation is crucial for robust image restoration. *arXiv preprint arXiv:2406.07435*, 2024. 1

[4] Shashank Agnihotri, Steffen Jung, and Margret Keuper. CosPGD: an efficient white-box adversarial attack for pixel-wise prediction tasks. In *Proc. International Conference on Machine Learning (ICML)*, 2024. 2, 5, 13, 14, 15

[5] Shashank Agnihotri, Shashank Priyadarshi, Hendrik Sommerhoff, Julia Grabinski, Andreas Kolb, and Margret Keuper. Roll the dice: Monte carlo downsampling as a low-cost adversarial defence, 2024. 1

[6] Shashank Agnihotri, Julian Yuya Caspary, Luca Schwarz, Xinyan Gao, Jenny Schmalfuss, Andres Bruhn, and Margret Keuper. Flowbench: A robustness benchmark for optical flow estimation, 2025. 14

[7] Shashank Agnihotri, David Schader, Nico Sharei, Mehmet Ege Kaçar, and Margret Keuper. Are Synthetic Corruptions A Reliable Proxy For Real-World Corruptions? In *CVPR Workshop On Synthetic Data for Computer Vision*, 2025. 5, 7, 12, 13

[8] Muhammad Awais, Weiming Zhuang, Lingjuan Lyu, and Sung-Ho Bae. Frod: Robust object detection for free. *CoRR*, 2023. 2

[9] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. European Conference on Computer Vision (ECCV)*, pages 611–625, 2012. 13

[10] Weiqin Chuah, Ruwan Tennakoon, Reza Hoseinnezhad, and Alireza Bab-Hadiashar. Deep learning-based incorporation of planar constraints for robust stereo depth estimation in autonomous vehicle applications. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6654–6665, 2021. 2

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016. 12, 13

[12] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 4, 14

[13] Anurag Das, Yongqin Xian, Yang He, Bernt Schiele, and Zeynep Akata. Sp2 net for generalized zero-label semantic segmentation. In *DAGM German Conference on Pattern Recognition*, pages 235–249. Springer, 2021. 1

[14] Anurag Das, Yongqin Xian, Dengxin Dai, and Bernt Schiele. Weakly-supervised domain adaptive semantic segmentation with prototypical contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15434–15443, 2023. 1

[15] Yuexiong Ding, Ming Zhang, Jia Pan, Jinxing Hu, and Xiaowei Luo. Robust object detection in extreme construction conditions. *Automation in Construction*, 165:105487, 2024. 2

[16] Ziyi Dong, Pengxu Wei, and Liang Lin. Adversarially-aware robust object detector. In *European Conference on Computer Vision*, pages 297–313. Springer, 2022. 2

[17] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 4

[18] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 7

[19] Paul Gavrikov, Shashank Agnihotri, Margret Keuper, and Janis Keuper. How do training methods influence the utilization of vision models? In *NeurIPS 2024 workshop on Interpretable AI: Past, Present and Future*, 2024. 7

[20] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018. 1

[21] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. 1

[22] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. International Conference on Learning Representations (ICLR)*, 2015. 2, 13, 14

[23] Julia Grabinski, Paul Gavrikov, Janis Keuper, and Margret Keuper. Robust models are less over-confident. *NeurIPS*, 2022. 1

[24] Julia Grabinski, Steffen Jung, Janis Keuper, and Margret Keuper. Frequencylowcut pooling-plug and play against catastrophic overfitting. In *European Conference on Computer Vision*, pages 36–57. Springer, 2022. 1

[25] Julia Grabinski, Janis Keuper, and Margret Keuper. Aliasing and adversarial robust generalization of cnns. *Machine Learning*, pages 1–27, 2022. 1

[26] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3273–3282, 2019. 1, 5

[27] Xianda Guo, Chenming Zhang, Juntao Lu, Yiqi Wang, Yiqun Duan, Tian Yang, Zheng Zhu, and Long Chen. Openstereo:

A comprehensive benchmark for stereo matching and strong baseline. *arXiv preprint arXiv:2312.00343*, 2023. 2, 7

[28] Xianda Guo, Chenming Zhang, Dujun Nie, Wenzhao Zheng, Youmin Zhang, and Long Chen. Lightstereo: Channel boost is all your need for efficient 2d cost aggregation. *arXiv preprint arXiv:2406.19833*, 2024. 7

[29] Xianda Guo, Chenming Zhang, Youmin Zhang, Dujun Nie, Ruilin Wang, Wenzhao Zheng, Matteo Poggi, and Long Chen. Stereo anything: Unifying stereo matching with large-scale mixed data. *arXiv preprint arXiv:2411.14053*, 2024. 7, 13

[30] Himanshu Gupta, Oleksandr Kotlyar, Henrik Andreasson, and Achim J Lilienthal. Robust object detection in challenging weather conditions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 7523–7532, 2024. 2

[31] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. International Conference on Learning Representations (ICLR)*, 2019. 1, 2, 12, 13, 14

[32] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. 1, 4

[33] J Hoffmann, S Agnihotri, Tonmoy Saikia, and Thomas Brox. Towards improving robustness of compressed cnns. In *ICML Workshop on Uncertainty and Robustness in Deep Learning (UDL)*, 2021. 1, 4

[34] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18963–18974, 2022. 13

[35] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *Proc. International Conference on Learning Representations (ICLR)*, 2017. 2, 13, 14

[36] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018. 2, 13, 14

[37] Yumeng Li, Dan Zhang, Margret Keuper, and Anna Khoreva. Intra-source style augmentation for improved domain generalization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 509–519, 2023. 1

[38] Yumeng Li, Margret Keuper, Dan Zhang, and Anna Khoreva. Adversarial supervision makes layout-to-image diffusion models thrive. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[39] Zhaoshuo Li, Xingtong Liu, Nathan Drenkow, Andy Ding, Francis X. Creighton, Russell H. Taylor, and Mathias Unberath. Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6197–6206, 2021. 1, 5, 13

[40] Zihua Liu, Yizhou Li, and Masatoshi Okutomi. Global occlusion-aware transformer for robust stereo matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3535–3544, 2024. 2

[41] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016. 1, 2, 4, 13

[42] Tejaswini Medi, Julia Grabinski, and Margret Keuper. Towards class-wise robustness analysis. *arXiv preprint arXiv:2411.19853*, 2024. 1

[43] Tejaswini Medi, Steffen Jung, and Margret Keuper. Fair-tat: Improving model fairness using targeted adversarial training. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 7827–7836. IEEE, 2025. 1

[44] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015. 2, 4, 13

[45] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019. 2

[46] Jan Müller, Reuben Docea, Matthias Hardner, Katja Krug, Paul Riedel, and Ronald Tetzlaff. Fast high-resolution disparity estimation for laparoscopic surgery. In *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 573–577. IEEE, 2022. 2

[47] Adarsh Prasad. *Towards Robust and Resilient Machine Learning*. PhD thesis, Carnegie Mellon University, 2022. 1

[48] Dimitrios Psychogyios, Evangelos Mazomenos, Francisco Vasconcelos, and Danail Stoyanov. Msdesis: Multitask stereo disparity estimation and surgical instrument segmentation. *IEEE transactions on medical imaging*, 41(11):3218–3230, 2022. 2

[49] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. *arXiv preprint arXiv:2005.03778*, 2020. 7

[50] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 5, 12, 13

[51] Erik Scheurer, Jenny Schmalfuss, Alexander Lis, and Andrés Bruhn. Detection defenses: An empty promise against adversarial patch attacks on optical flow. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024. 1

[52] Jenny Schmalfuss, Lukas Mehl, and Andrés Bruhn. Attacking motion estimation with adversarial snow. In *Proc. ECCV Workshop on Adversarial Robustness in the Real World (AROW)*, 2022.

[53] Jenny Schmalfuss, Philipp Scholze, and Andrés Bruhn. A perturbation-constrained adversarial attack for evaluating the robustness of optical flow. In *Proc. European Conference on Computer Vision (ECCV)*, pages 183–200, 2022. 1, 5

[54] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. 7

[55] Zhelun Shen, Yuchao Dai, and Zhibo Rao. Cfnet: Cascade and fused cost volume for robust stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13906–13915, 2021. 1, 5, 13

[56] Hendrik Sommerhoff, Shashank Agnihotri, Mohamed Saleh, Michael Moeller, Margret Keuper, and Andreas Kolb. Differentiable sensor layouts for end-to-end learning of task-specific camera parameters. *arXiv preprint arXiv:2304.14736*, 2023. 1

[57] Hendrik Sommerhoff, Shashank Agnihotri, Mohamed Saleh, Michael Moeller, Margret Keuper, Bhaskar Choubey, and Andreas Kolb. Task driven sensor layouts-joint optimization of pixel layout and network parameters. In *2024 IEEE International Conference on Computational Photography (ICCP)*, pages 1–10. IEEE, 2024. 1

[58] Shiyu Tang, Ruihao Gong, Yan Wang, Aishan Liu, Jiakai Wang, Xinyun Chen, Fengwei Yu, Xianglong Liu, Dawn Song, Alan Yuille, Philip H.S. Torr, and Dacheng Tao. Robustart: Benchmarking robustness on architecture design and training techniques. *https://arxiv.org/pdf/2109.05211.pdf*, 2021. 2

[59] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. *ArXiv*, abs/2001.03994, 2020. 2, 5

[60] Bo Yang, Siyuan Xu, Lirong Yin, Chao Liu, and Wenfeng Zheng. Disparity estimation of stereo-endoscopic images using deep generative network. *ICT Express*, 2024. 2

[61] Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2d feature representations by 3d-aware fine-tuning. In *European Conference on Computer Vision*, pages 57–74. Springer, 2024. 1

[62] Jiawei Zhang, Jiahe Li, Lei Huang, Xiaohan Yu, Lin Gu, Jin Zheng, and Xiao Bai. Robust synthetic-to-real transfer for stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20247–20257, 2024. 2

[63] Shuangli Zhang, Weijian Xie, Guofeng Zhang, Hujun Bao, and Michael Kaess. Robust stereo matching with surface normal prediction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2540–2547, 2017. 2

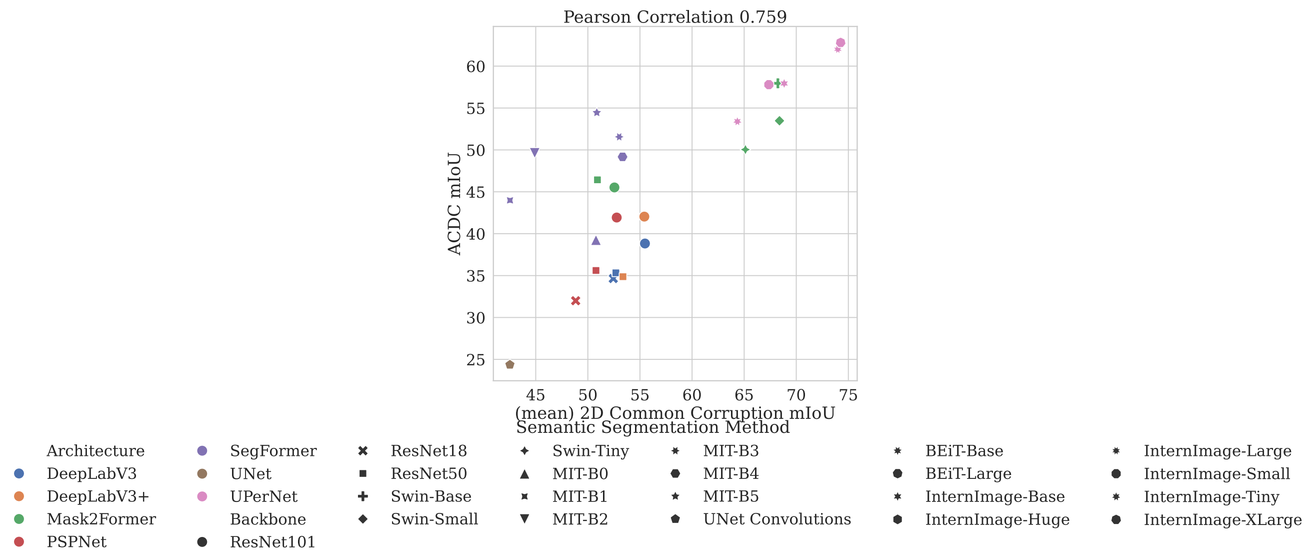

Figure 7. Results from work by Agnihotri et al. [7]. Here they find a **very strong positive correlation between mean mIoU over the ACDC evaluation dataset [50] and mean mIoU over each 2D Common Corruption [31]** over the Cityscapes dataset [11]. All models were trained using the training subset of the Cityscapes dataset. ACDC is the Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding captured in similar scenes are cityscapes but under four different domains: Day/Night, Rain, Snow, and Fog in the wild. ACDC is a community-used baseline for evaluating the performance of semantic segmentation methods on domain shifts observed in the wild.

DISPBENCH: Benchmarking Disparity Estimation to Synthetic Corruptions

Paper #6 Supplementary Material

Table Of Content

The supplementary material covers the following information:

- Appendix A: We show that synthetic 2D Common Corruptions indeed serve as a proxy to domain shifts in the real world.
- Appendix B: Additional implementation details for the evaluated benchmark.
- Appendix C: In detail description of the attacks.
- Appendix D: DISPBENCH function call to get model weights.
- Appendix E: In detail explanation of the available functionalities of the DISPBENCH benchmarking tool and description of the arguments for each function.
- Appendix F: Here we provide additional results from the benchmark evaluated using DISPBENCH.

Please note that due to the similarity of the objective, many aspects of this appendix are very similar to Agnihotri et al. [7].

A. Do Synthetic Corruptions Represent The Real World?

In their work Agnihotri et al. [7], they find the correlation between mean mIoU over the ACDC evaluation dataset [50] and mean mIoU over each 2D Common Corruption [31] over the Cityscapes dataset [11]. We include Figure 7 from their work here for ease of understanding. All models were trained using the training subset of the Cityscapes dataset. ACDC is the Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding captured in similar scenes are cityscapes but under four different domains: Day/Night, Rain, Snow, and Fog in the wild. ACDC is a community-used baseline for evaluating the performance of semantic segmentation methods on domain shifts observed in the wild. They find that there exists a very strong positive correlation between the two. This shows, that **yes, synthetic corruptions can serve as a proxy for the real world**. Unfortunately, a similar "in the wild" captured dataset does not exist for optical flow estimation to evaluate the effect of domain shifts on the performance of optical flow methods. However, given that for the task of semantic segmentation, we find a very high positive correlation between the performance on real-world corruptions and synthetic corruptions, it is a safe assumption that the same would hold true for optical flow estimation as well. Thus, in this work, we evaluate against synthetic 2D Common Corruptions [31] and synthetic 3D Common Corruptions [34].

B. Implementation Details Of The Benchmark

Following, we provide details regarding the experiments done for creating the benchmark used in the analysis.

Compute Resources. Most experiments were done on a single 40 GB NVIDIA Tesla V100 GPU each, however, MS-RAFT+, FlowFormer, and FlowFormer++ are more compute-intensive, and thus 80GB NVIDIA A100 GPUs or NVIDIA H100 were used for these models, a single GPU for each experiment.

Datasets Used. Performing adversarial attacks and OOD robustness evaluations are very expensive and compute-intensive. Thus, performing evaluation using all model-dataset pairs is not possible given the limited computing resources at our disposal. Thus, for the benchmark, we only use FlyingThings3D and KITTI2015, as these are the most commonly used datasets for evaluation [29, 39, 41, 55].

Metrics Calculation. In this work, for robustness evaluations we consider the mC-EPE, which is the mean End-Point-Error of a method, against common corruptions at a given severity, over every input image from the validation dataset. We use all 15 2D Common Corruptions: 'Gaussian Noise', Shot Noise', 'Impulse Noise', 'Defocus Blur', 'Frosted Glass Blur', 'Motion Blur', 'Zoom Blur', 'Snow', 'Frost', 'Fog', 'Brightness', 'Contrast', 'Elastic Transform', 'Pixelate', 'JPEG Compression'. All the common corruptions are at severity=$\{1, 2, 3, 4, 5\}$. [34] offers more 3D Common Corruptions, however computing them is resource intensive. Thus, given our limited resources and an overlap in the corruptions between 2D Common Corruptions and 3D Common Corruptions, we focus on generating 3D Common Corruptions for now, however, we intend to extend DispBench to also evaluate on the 3D Common Corruptions.

Calculating the EPE. EPE is the Euclidean distance between the two vectors, where one vector is the predicted flow by the disparity estimation method and the other vector is the ground truth in case of i.i.d. performance evaluations, non-targeted attacks evaluations, and OOD robustness evaluations, while it is the target flow vector, in case of targeted attacks. For each dataset, the EPE value is calculated over all the samples of the evaluation set of the respective dataset and then the mean EPE value is used as the mean-EPE of the respective method over the respective dataset.

C. Description of DispBench

Following, we describe the benchmarking tool, DispBench. There exists no standardized tool for evaluating the performance of disparity estimation methods. Thus, the codebase for such a tool had to be written from scratch. In the following, we describe the benchmarking tool, DispBench. Currently it supports 4 unique architectures (new architectures to be added to DispBench with time) and 3 distinct datasets, namely FlyingThings3D [41], KITTI2015 [44], MPI Sintel [9] (clean and final) (please refer Sec. 2.4 for additional details on the datasets). It enables training and evaluations on all aforementioned datasets including evaluations using SotA adversarial attacks such as CosPGD [4], and other commonly used adversarial attacks like BIM [36], PGD [35], FGSM [22], under various Lipshitz (l_p) norm bounds. Additionally, it enables evaluations for Out-of-Distribution (OOD) robustness by corrupting the inference samples using 2D Common Corruptions [31].

Following we show the basic commands to use DISPBENCH. We describe each attack and common corruption supported by DISPBENCH in detail in Appendix C. It enables training and evaluations on all aforementioned datasets including evaluations using SotA adversarial attacks such as CosPGD [4], and other commonly used adversarial attacks like BIM [36], PGD [35], FGSM [22], under various lipshitz (l_p) norm bounds. Additionally, it enables evaluations for Out-of-Distribution (OOD) robustness by corrupting the inference samples using 2D Common Corruptions [31].

We follow the nomenclature set by RobustBench [12] and use "threat_model" to define the kind of evaluation to be performed. When "threat_model" is defined to be "None", the evaluation is performed on unperturbed and unaltered images, if the "threat_model" is defined to be an adversarial attack, for example "PGD", "CosPGD" or "PCFA", then DISPBENCH performs an adversarial attack using the user-defined parameters. Whereas, if "threat_model" is defined to be "2DCommonCorruptions" or "3DCommonCorruptions", the DISPBENCH performs evaluations after perturbing the images with 2D Common Corruptions and 3D Common Corruptions respectively.

If the queried evaluation already exists in the benchmark provided by this work, then DISPBENCH simply retrieves the evaluations, thus saving computation.

C.1. Adversarial Attacks

Please note that due to the similarity of the objective, many aspects of this appendix are very similar to Agnihotri et al. [6]. DISPBENCH enables the use of many white-box adversarial attacks to help users better study the reliability of their disparity methods. We choose to specifically include these white-box adversarial attacks as they either serve as the common benchmark for adversarial attacks in classification literature (FGSM, BIM, PGD, APGD) or they are unique attacks proposed specifically for pixel-wise prediction tasks (CosPGD). These attacks can either be *Non-targeted* which are designed to simply fool the model into making incorrect predictions, irrespective of what the model eventually predicts, or can be *Targeted*, where the model is fooled to make a certain prediction. Most attacks can be, designed to be either Targeted or Non-targeted, these include, FGSM, BIM, PGD, APGD, CosPGD, and Adversarial Weather. In our current implementation, we are limited to Non-targeted attacks. Following, we discuss these attacks in detail and highlight their key differences.

FGSM. Assuming a non-targeted attack, given a model f_θ and an unperturbed input sample $\boldsymbol{X}^{\text{clean}}$ and ground truth label $\boldsymbol{Y}$, FGSM attack adds noise δ to $\boldsymbol{X}^{\text{clean}}$ as follows,

$$\boldsymbol{X}^{\text{adv}} = \boldsymbol{X}^{\text{clean}} + \alpha \cdot \text{sign}\nabla_{\boldsymbol{X}^{\text{clean}}} L(f_\theta(\boldsymbol{X}^{\text{clean}}), \boldsymbol{Y}), \tag{1}$$

$$\delta = \phi^\epsilon(\boldsymbol{X}^{\text{adv}} - \boldsymbol{X}^{\text{clean}}), \tag{2}$$

$$\boldsymbol{X}^{\text{adv}} = \phi^r(\boldsymbol{X}^{\text{clean}} + \delta). \tag{3}$$

Here, $L(\cdot)$ is the loss function (differentiable at least once) which calculates the loss between the model prediction and ground truth, $\boldsymbol{Y}$. α is a small value of ϵ that decides the size of the step to be taken in the direction of the gradient of the loss w.r.t. the input image, which leads to the input sample being perturbed such that the loss increases. $\boldsymbol{X}^{\text{adv}}$ is the adversarial sample obtained after perturbing $\boldsymbol{X}^{\text{clean}}$. To make sure that the perturbed sample is semantically indistinguishable from the unperturbed clean sample to the human eye, steps from Eq. (2) and Eq. (3) are performed. Here, function ϕ^ϵ is clipping the δ in ϵ-ball for ℓ_∞-norm bounded attacks or the ϵ-projection in other l_p-norm bounded attacks, complying with the ℓ_∞-norm or other l_p-norm constraints, respectively. While function ϕ^r clips the perturbed sample ensuring that it is still within the valid input space. FGSM, as proposed, is a single step attack. For targeted attacks, $\boldsymbol{Y}$ is the target and α is multiplied by -1 so that a step is taken to minimize the loss between the model's prediction and the target prediction.

BIM. This is the direct extension of FGSM into an iterative attack method. In FGSM, $\boldsymbol{X}^{\text{clean}}$ was perturbed just once. While in BIM, $\boldsymbol{X}^{\text{clean}}$ is perturbed iteratively for time steps $t \in [0, \boldsymbol{T}]$, such that $t \in \mathbb{Z}^+$, where $\boldsymbol{T}$ are the total number of permissible attack iterations. This changes the steps of the attack from FGSM to the following,

$$\boldsymbol{X}^{\text{adv}_{t+1}} = \boldsymbol{X}^{\text{adv}_t} + \alpha \cdot \text{sign}\nabla_{\boldsymbol{X}^{\text{adv}_t}} L(f_\theta(\boldsymbol{X}^{\text{adv}_t}), \boldsymbol{Y}), \tag{4}$$

$$\delta = \phi^\epsilon(\boldsymbol{X}^{\text{adv}_{t+1}} - \boldsymbol{X}^{\text{clean}}), \tag{5}$$

$$\boldsymbol{X}^{\text{adv}_{t+1}} = \phi^r(\boldsymbol{X}^{\text{clean}} + \delta). \tag{6}$$

Here, at t=0, $\boldsymbol{X}^{\text{adv}_t}$=$\boldsymbol{X}^{\text{clean}}$.

PGD. Since in BIM, the initial prediction always started from $\boldsymbol{X}^{\text{clean}}$, the attack required a significant amount of steps to optimize the adversarial noise and yet it was not guaranteed that in the permissible ϵ-bound, $\boldsymbol{X}^{\text{adv}_{t+1}}$ was far from $\boldsymbol{X}^{\text{clean}}$. Thus, PGD proposed introducing stochasticity to ensure random starting points for attack optimization. They achieved this by perturbing $\boldsymbol{X}^{\text{clean}}$ with $\mathcal{U}(-\epsilon, \epsilon)$, a uniform distribution in $[-\epsilon, \epsilon]$, before making the first prediction, such that, at t=0

$$\boldsymbol{X}^{adv_t} = \phi^r(\boldsymbol{X}^{clean} + \mathcal{U}(-\epsilon, \epsilon)). \tag{7}$$

APGD. Auto-PGD is an effective extension to the PGD attack that effectively scales the step size α over attack iterations considering the compute budget and the success rate of the attack.

CosPGD. All previously discussed attacks were proposed for the image classification task. Here, the input sample is a 2D image of resolution $\mathrm{H} \times \mathrm{W}$, where H and W are the height and width of the spatial resolution of the sample, respectively. Pixel-wise information is inconsequential for image classification. This led to the pixel-wise loss $\mathcal{L}(\cdot)$ being aggregated to $\mathrm{L}(\cdot)$, as follows,

$$L(f_\theta(\boldsymbol{X}^{\text{adv}_t}), \boldsymbol{Y}) = \frac{1}{\mathrm{H} \times \mathrm{W}} \sum_{i \in \mathrm{H} \times \mathrm{W}} \mathcal{L}(f_\theta(\boldsymbol{X}^{\text{adv}_t})_i, \boldsymbol{Y}_i). \tag{8}$$

This aggregation of $\mathcal{L}(\cdot)$ fails to account for pixel-wise information available in tasks other than image classification, such as pixel-wise prediction tasks like Optical Flow estimation, and disparity estimation. Thus, in their work [4] propose an effective extension of the PGD attack that takes pixel-wise information into account by scaling $\mathcal{L}(\cdot)$ by the alignment between the distribution of the predictions and the distributions of $\boldsymbol{Y}$ before aggregating leading to a better-optimized attack, modifying Eq. (4) as follows,

$$\boldsymbol{X}^{\text{adv}_{t+1}} = \boldsymbol{X}^{\text{adv}_t} + \alpha \cdot \text{sign}\nabla_{\boldsymbol{X}^{\text{adv}_t}} \sum_{i \in H \times W} \cos\left(\psi(f_\theta(\boldsymbol{X}^{\text{adv}_t})_i), \Psi(\boldsymbol{Y}_i)\right) \cdot \mathcal{L}\left(f_\theta(\boldsymbol{X}^{\text{adv}_t})_i, \boldsymbol{Y}_i\right). \tag{9}$$

Where, functions ψ and Ψ are used to obtain the distribution over the predictions and $\boldsymbol{Y}_i$, respectively, and the function $\cos$ calculates the cosine similarity between the two distributions. CosPGD is the unified SotA adversarial attack for pixel-wise prediction tasks.

In Figure 2, we show examples of adversarial attacks, on STTR using the KITTI2015 dataset. We show the samples before and after the attacks and the predictions before and after the respective adversarial attacks.

D. Model Zoo

It is challenging to find all checkpoints whereas training them is time and compute-exhaustive. Thus we gather available model checkpoints made available online by the respective authors. The trained checkpoints for all models available in DISPBENCH can be obtained using the following lines of code:

```
from dispbench.evals import load_model
model = load_model(model_name='STTR',
    dataset='KITTI2015')
```

Each model checkpoint can be retrieved with the pair of 'model_name', the name of the model, and 'dataset', the dataset for which the checkpoint was last finetuned.

E. DISPBENCH Usage Details

Following we provide a detailed description of the evaluation functions and their arguments provided in DISPBENCH.

E.1. Adversarial Attacks

To evaluate a model for a given dataset, on an attack, the following lines of code are required.

```
from dispcbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015' retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model, for example, when the threat model is a PGD attack, 'config.yml' could contain 'threat_model=*"PGD"*', 'iterations=*20*', 'alpha=*0.01*', 'epsilon=*8*', and 'lp_norm=*"Linf"*'. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False' then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the adversarial attack to be used, given as a string.
  - 'iterations' are the number of attack iterations, given as an integer.
  - 'epsilon' is the permissible perturbation budget ϵ given a floating point (float).
  - 'alpha' is the step size of the attack, α, given as a floating point (float).
  - 'lp_norm' is the Lipschitz continuity norm (l_p-norm) to be used for bounding the perturbation, possible options are 'Linf' and 'L2' given as a string.
  - 'target' is false by default, but to do targeted attacks, either the user can set 'target'=True, to use the default target of $\overrightarrow{0}$, or can pass a specific tensor to be used as the target.

E.2. 2D Common Corruptions

To evaluate a model for a given dataset, with 2D Common Corruptions, the following lines of code are required.

```
from dispbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015', retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model; for example, when the threat model is 2D Common Corruption, 'config.yml' could contain 'threat_model=*"2DCommonCorruption"*', and 'severity=*3*'. Please note, when the 'threat_model' is a common corruption type, DISPBENCH performs evaluations on all corruptions under the respective 'threat_model' and returns the method's performance on each corruption at the requested severity. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False', then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the common corruption to be used, given as a string, i.e. '2DCommonCorruption'.
  - 'severity' is the severity of the corruption, given as an integer between 1 and 5 (both inclusive).

DISPBENCH supports the following 2D Common Corruption: 'gaussian_noise', shot_noise', 'impulse_noise', 'defocus_blur', 'frosted_glass_blur', 'motion_blur', 'zoom_blur', 'snow', 'frost', 'fog', 'brightness', 'contrast', 'elastic', 'pixelate', 'jpeg'. For the evaluation, DISPBENCH will evaluate the model on the validation images from the respective dataset corrupted using each of the aforementioned corruptions for the given severity, and then report the mean performance over all of them.

F. Extension To Analysis

Following, we extend the analysis from Section 4 and report additional evaluations from DISPBENCH.

F.1. KITTI2015

Following, we provide evaluations of on the KITTI2015 dataset. In Figure 8 we report the evaluations of all considered adversarial attacks with $\epsilon = \frac{8}{255}$ and α=0.01 under the ℓ_∞-norm bound using the KITTI2015 validation dataset.

In Figure 9, we extend the evaluations from Figure 6, showing that the observations made over the mean performance over all corruptions also hold for every individual corruption.

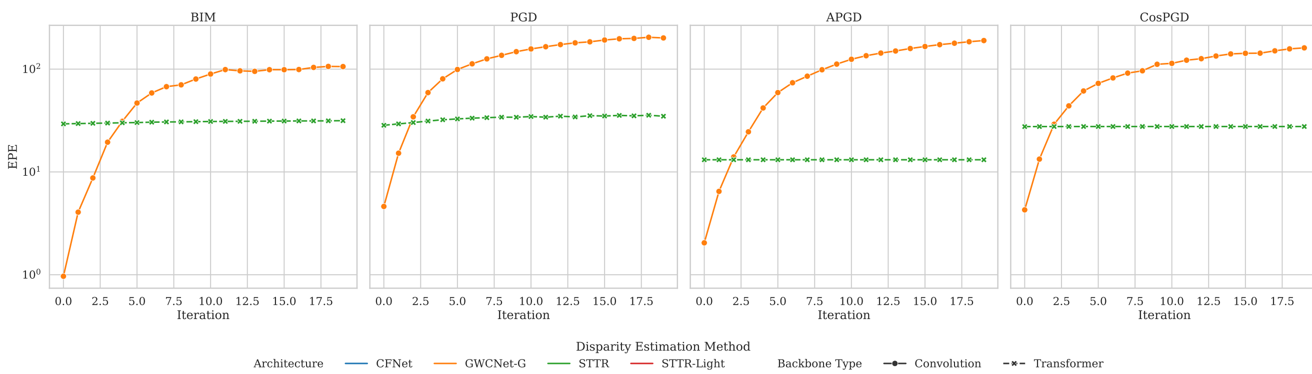

Figure 8. Evaluations of all considered adversarial attacks with $\epsilon = \frac{8}{255}$ and α=0.01 under the ℓ_∞-norm bound using the KITTI2015 validation dataset.

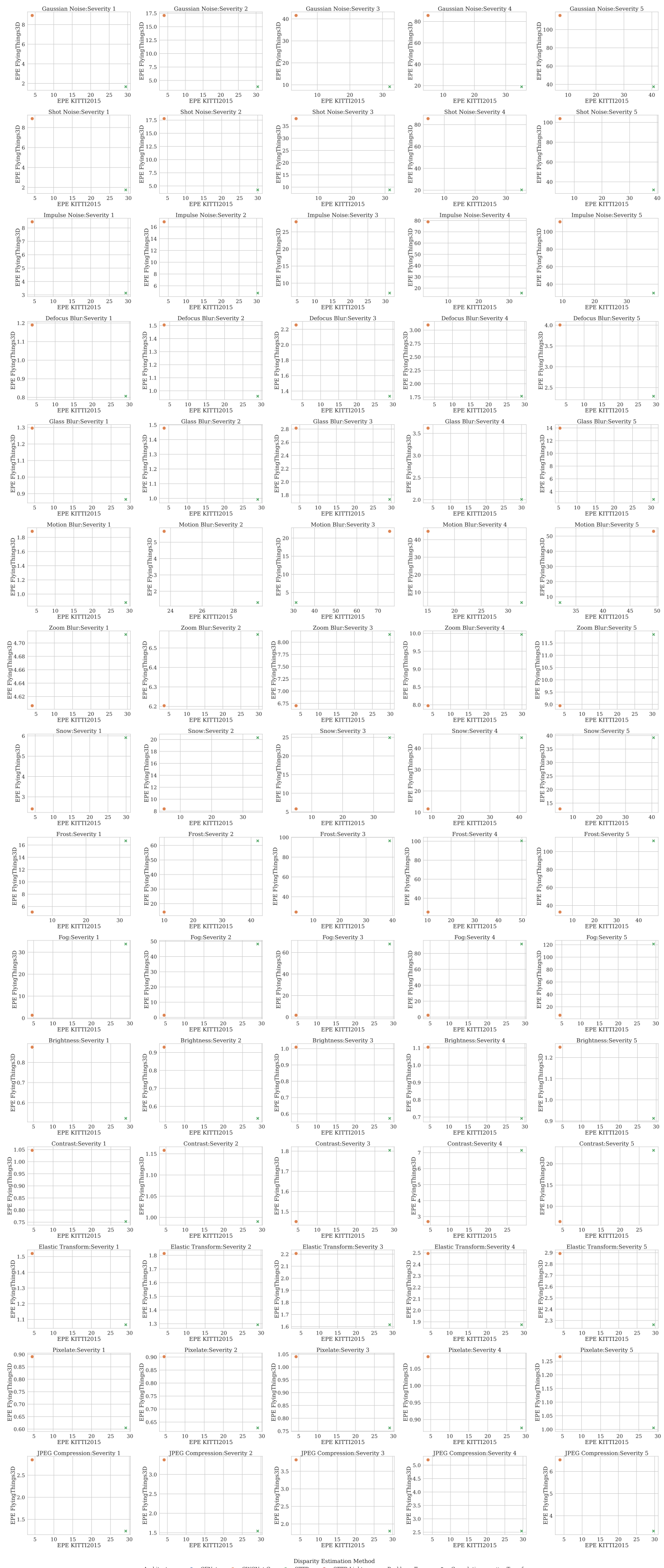

Figure 9. For the same architecture, we evaluate checkpoints pretrained on Flyingthings3D against synthetic 2D Common Corruption on Flyingthings3D and correlate its performance with the checkpoint trained on KITTI2015 against synthetic 2D Common Corruptions on KITTI2015. Here, we report the EPE across every individual corruption for a given severity level. We observe no correlation in performance, indicating that synthetic corruptions on synthetic datasets cannot be used as a proxy for real-world corruptions.