Computer Science

Monocular Depth Estimation Neural Networks: a Comprehensive Analysis

Rutger van der Beek

Supervisors:
Prof. dr. Michael S.K. Lew
Dr. Erwin M. Bakker

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

23/03/2023

Abstract

In this thesis, we analyse three state-of-the-art monocular depth estimation methods. These neural networks are used for various Computer Vision tasks, such as autonomous driving. We seek to discover the strengths and weaknesses of these methods in order to get a deeper understanding of them. For this, we construct seven categories of scenes and analyse the results of the three methods in these categories. For the analysis, we use eight commonly used metrics and a new metric proposed by us.

Contents

1 Introduction

Monocular depth estimation is a branch of Computer Vision and is important for, for example, autonomous driving, robotics, and augmented reality. However, as many 3D scenes can be constructed from the same 2D image, it is an ill-posed problem. Notwithstanding this, different methods using deep convolutional neural networks have recently been able to accurately estimate depth maps using only single images as input. Among these methods are AdaBins [1], Big to Small [8], and Virtual Normal [17]. These three state-of-the-art methods have good results and each has its own innovative way to try to better estimate the depth maps.

In this thesis, we will look at these three methods and extensively analyse them using different image categories to ultimately try to learn what the strengths and weaknesses of these methods are and why. With these results, we aim to get a better picture of the methods which we do not get with their current presentations. Most often new methods only compare themselves to other methods based on the same datasets on which they were trained. These subsets are thus similar to the training data which can be a problem as it remains unknown how well the methods will perform in real settings. By constructing a testing dataset from one or more different datasets, we will get a better idea of their general performance.

In depth estimation, KITTI [5] and NYU Depth V2 [9] are the most popular datasets for training and testing. As the three methods discussed in this thesis use these datasets for training as well, we will not use them for testing. Rather, we will use the SUN RGB-D [10, 13, 7, 9] and DIODE [12] datasets, two publicly available datasets with a plethora of different scenes. For the analysis we will use eight evaluation metrics: three different thresholds for the accuracy, the root mean squared error and its logarithmic variant, the absolute relative error, the squared relative error, and the log error. In addition to these metrics, which are all used in previous works, we propose a new metric, the local shape threshold, as the ninth metric.

One notion that will not be discussed in depth in this thesis, but is important for the subject, is the general workings of Neural Networks. These networks require massive amounts of training data with the corresponding ground truth, which is naturally relatively hard to obtain. This means that improvements in datasets will likely greatly influence the performance of methods on general scenes. Yin et al. [18] describe this more in-depth and have their own attempt at solving the, as they call it, generalisation issue of monocular depth estimation.

In section 2 we will give a more in-depth explanation of the workings of the three methods. Section 3 describes the way the experiments are performed, together with an explanation of the datasets. Section 4 presents the results of the experiments. In section 5 we discuss and share our insights on the results. Lastly, in section 6 we will draw a conclusion.

This thesis has been written for the bachelor Computer Science at LIACS of Leiden University under the supervision of Michael Lew and Erwin Bakker.

2 Researched Methods

Depth estimation is a relatively young field of research and there are plenty of new methods released every year. For this thesis, we have chosen three state-of-the-art methods published in the last four years with code publicly available. These methods each use a different approach to deliver an estimation of the depth map.

2.1 AdaBins

AdaBins [1] is the most recent method discussed in this thesis. Published in 2021, it uses the novel idea of adaptive bins to better estimate depth with the use of different depth ranges. AdaBins uses a standard encoder-decoder network followed by their own addition, as seen in Figure 1. They have chosen to implement this module after the decoder to improve the resolution by working on the original resolution instead of a lower one, which would have been the case if the module was placed between the encoder and decoder. The idea behind their implementation is to divide the depth range of every individual image into bins calculated for that picture specifically, instead of a set of bins trained on an entire dataset which remains the same for all images, as was the case for DORN [4]. These adaptive bins increase the amount of depth intervals at the depth range most present in an image, which by itself decreases the amount of depth discretisation artifacts in those areas. This is further decreased by calculating the final depth prediction with the linear combination of the Softmax of a pixel together with the center of the depth bin assigned to that pixel, resulting in a smooth depth estimation. For the loss function, AdaBins uses the Training Loss as introduced by Eigen et al. [3] and a bin-center density loss, which tries to ensure the depth ranges for the bins are estimated optimally.

Of these three methods, AdaBins has the best performance on the NYU Depth V2 [9] and KITTI [5] datasets when trained on them.

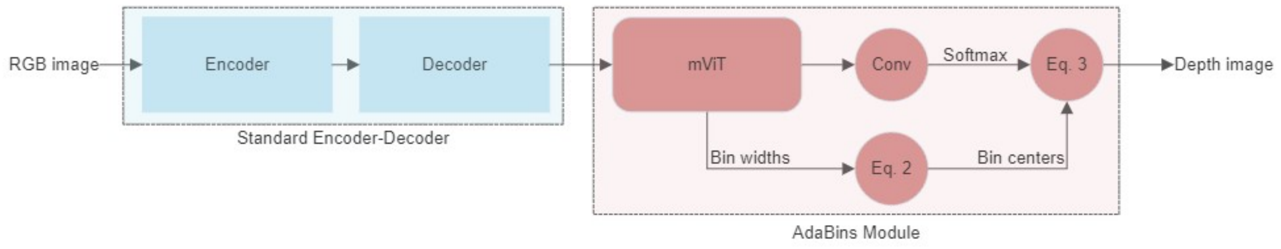

Figure 1: AdaBins pipeline [1]

2.2 Big to Small

Big to Small (BTS) [8], as the name suggests, shrinks the input to create an estimation of the depth. It uses the standard encoder-decoder network with a dense feature extractor, a contextual feature extractor (ASPP [15]), and their addition, the Local Planar Guidance layers. These layers use a lower resolution than the original image, 8, 4, and 2 times smaller, but do have an output of the original size as well, ensuring they can be passed through to later steps. This pipeline can be seen in Figure 2. Together with a reduction layer, to better estimate the object borders, the three outputs are non-linearly combined to obtain the final depth estimation. For example, the coarsest outputs can be the base of the estimation, while the finer outputs fill in the details. By using the outputs of these layers instead of nearest neighbor upsampling and conventional skip connections, an estimation of higher resolution can be obtained. BTS uses the training loss as introduced by Eigen et al. [3].

Based on the paper BTS performs well on discerning objects, which can be attributed to the reduction layer.

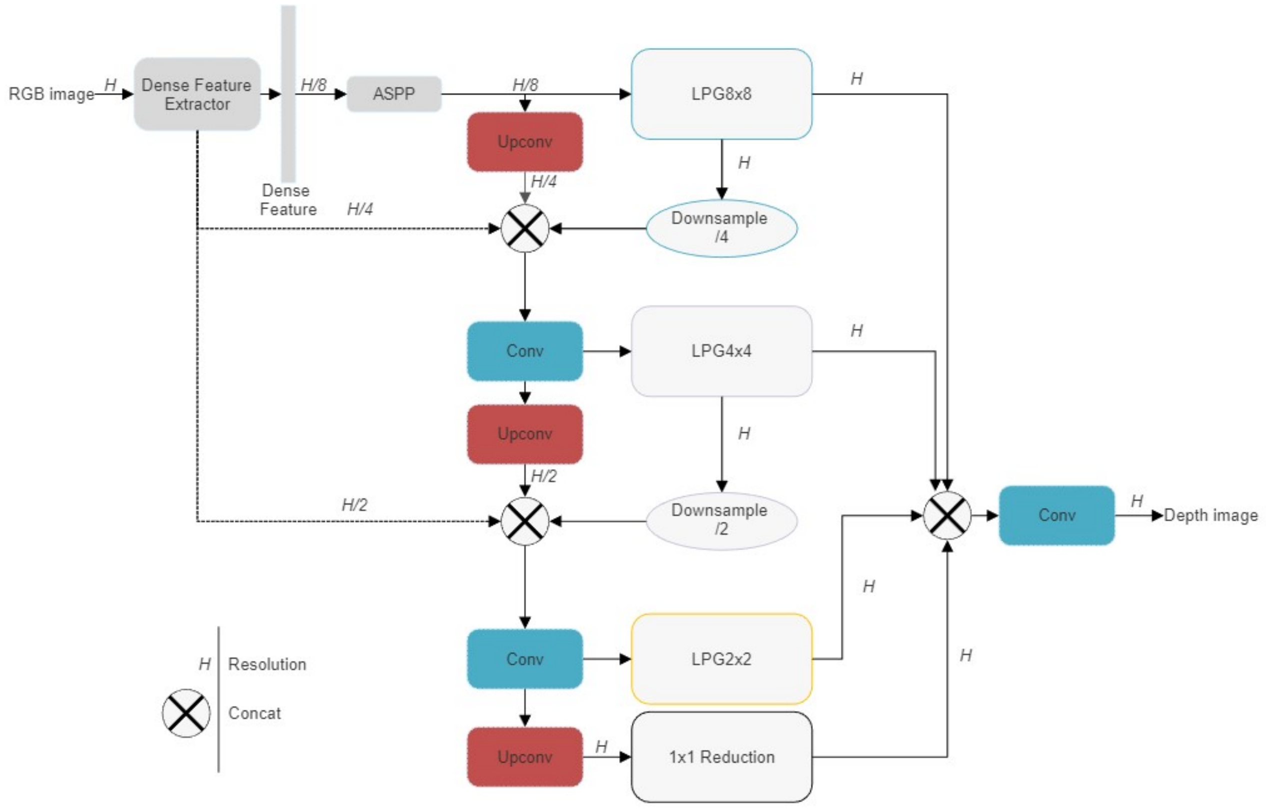

Figure 2: BTS pipeline [8]

2.3 Virtual Normal

Virtual Normal Loss (VNL) [17, 16] uses an encoder-decoder network to predict the depth map of an image. This depth map is subsequently transformed into a point cloud and this point cloud is compared to the point cloud of the ground truth using the virtual normal, which is calculated with the camera coordinate as the world coordinate. This pipeline can be seen in Figure 3. By using this virtual normal instead of the surface normal, VNL compares the depth maps on a global level instead of a local level, thereby being more resilient to noise. The divergence of multiple of these virtual normals is used as one of the loss functions of the method. VNL also makes use of a second pre-existing constraint, the weighted cross-entropy loss, as seen in [2], a pixel-wise loss that classifies the depth estimation as a classification problem.

VNL is noteworthy due to its direct computation of the point cloud, which can greatly help with calculating other 3D features instead of using a different submodel for this.

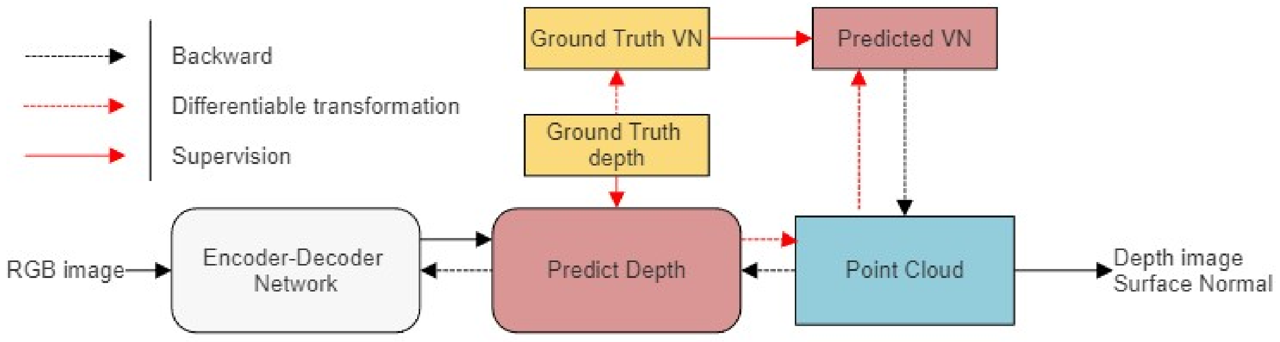

Figure 3: VNL pipeline [16]

3 Experiments

3.1 Datasets

Table 1 contains a concise overview of the datasets. The rest of this subsection will further elaborate on them.

Dataset	Sensor	Indoors/Outdoors	Scenes	Images
NYU Depth V2 [9]	Microsoft Kinect	Indoors	464	407K
SUN RGB-D [10, 13, 7, 9]	Multiple[1]	Indoors	47	10K
DIODE [12]	FARO Focus S350	Indoors and Outdoors	25	26K
KITTI [5]	Lidar	Outdoors	61	93K

Table 1: Overview of the Datasets

3.1.1 Training

For this thesis, the trained models found on the GitHub pages of the methods are used. For AdaBins, it is trained with EfficientNet-B5 [11] as the base network, for BTS DenseNet161 [6], and for VNL ResNeXt101 [14]. These base networks are the same for both the models trained on KITTI and the models trained on NYU Depth V2.

[1]Intel Realsense, Asus Xtion, Kinect v1, Kinect v2

The choice was made to take the best-performing models for each method instead of trying to keep the base networks the same for each method, because the base networks used in the different methods did not fully overlap and because we are mainly looking at the strengths and weaknesses of each method individually. This does mean that this difference should be kept in mind when comparisons between methods are made.

We use both the network trained on KITTI and the network trained on NYU Depth V2 because we want to find the strengths and weaknesses of the methods and not those of the training data. These two training datasets differ enough to get a better idea.

KITTI [5]**.** The KITTI dataset is a large outdoor dataset containing images taken with a video camera and depth maps obtained using a Lidar while driving through the streets of Karlsruhe. This means the dataset only contains the street view of one city, which causes it to lack somewhat in diversity, but makes it a decent dataset for training neural networks used for self-driving cars, for example. In total it has 61 different scenes from the city roads. It is often used as a training and testing dataset by methods for depth estimation to compare to other methods. VNL and BTS train on about 23.5K images from 32 scenes and AdaBins trains on a subset of about 26K images

NYU Depth V2 [9]**.** The NYU Depth V2 dataset is a large indoor dataset with images and depth maps taken in houses and commercial buildings in three American cities using a Microsoft Kinect. The dataset has some variety, but larger indoor areas like sports halls are notably absent, which is perhaps because of the range at which the Kinect can detect the depth. It has 464 different scenes of which VNL uses 249, with a total of 29K images, for training. BTS trains on about 24K images from 249 scenes and AdaBins on 50K images.

3.1.2 Testing

For the testing, we manually constructed several categories using the DIODE and SUN RGB-D datasets. These categories were chosen based on different types of shapes and differences in depth cues. With four indoor and three outdoor categories, we hope to be able to highlight the main strengths and weaknesses of the methods. For BTS we used the PyTorch implementation, the other methods only have one implementation. Furthermore, during the testing, some minor changes were made to the three methods to ensure the output was homogenised. For testing, the methods require a depth range in which they place their estimations. However, as this depth range only scales the output to the depth range and does not affect the results, we chose to use the standard max depth for NYU Depth V2, which is 10 meters.

Other than these exceptions above, the testing was performed as is standard for the research community, which is the same as used in, for example, Yin et al. [17].

DIODE [12]**.** DIODE is a diverse dataset with indoor and outdoor images together with the corresponding depth maps obtained using a FARO Focus S350 scanner. This laser scanner can be used indoors and outdoors and has a dense output resulting in rather accurate depth maps. It also has a validity mask file for each depth map, which has the valid and invalid depth pixels as true or false respectively that can be used to only evaluate the valid pixels. DIODE has 25 scenes publicly available with a total of 9K indoor and 17K outdoor images.

SUN RGB-D [10]. In addition to its own images, the SUN RGB-D dataset also contains images from the NYU Depth V2 [9], B3DO [7], and SUN3D [13] datasets, although for our testing we avoid the NYU Depth V2 images of this dataset. All images in this dataset are from indoor scenes and are not greatly varied. The images are captured with the following sensors: the Intel Realsense, the Asus Xtion, the Kinect v1, and the Kinect v2. Obtained depth maps are then improved by combining multiple frames of a video into a single depth map. SUN RGB-D has 47 scenes with a total of 10K images.

3.2 Data Categories

- **Close-ups:** This indoor category consists of 101 images taken from a short distance, resulting in a small depth range. This helps to evaluate the method's ability to cope with and estimate finer details.

- **Longer distance:** This indoor category consists of 104 images with objects further away, resulting in a larger depth range. This can help with discerning whether the method performs well with less detail in parts of the image and occlusion of objects.

- **Fake-light source:** This indoor category consists of 127 images with turned-on sources of a synthetic light in the image, which causes the source to be obscured with a bright light. This category aims to evaluate a method's ability to cope with bright unnatural spotlighting instead of the usual sunlight or less bright, more evenly distributed unnatural lighting.

- **Plants indoors:** This indoor category consists of 71 images containing plants together with non-animate objects. Indoor data with only, or mostly, plants proved to be difficult to obtain, so these images were chosen instead. As plants have more unique shapes with more irregularities, this category might help to identify whether the method can perform well on those shapes and object types.

- **Forest:** This outdoor category consists of 152 images taken within a forest. Trees have a rather different depth map compared to buildings and other man-made structures, which is why we thought it to be a good way to see how the methods behave on new completely different data with complex scenes.

- **Parks:** This outdoor category consists of 191 images with plants and man-made structures as generally found in parks. This can test methods on their ability to accurately estimate depth in scenes with varying objects, like benches and foliage, at varying distances.

- **Man-made structures:** This outdoor category consists of 125 images with only structures such as buildings in the scene. This category tests the methods' capability of handling geometric shapes, including different perspectives.

3.3 Normalising the Images

All input images were normalised to a resolution of 640x480. There are two ways to accomplish this, either by resizing the image so that one axis matches the target resolution and padding the other axis with black bars at both sides, or by cropping out the middle of the image, which is only possible if the original image is larger than the desired resolution. The first method scales all image data, so the resulting image still has the exact same scene. Although the two edges with bars will get some distorted predictions as the methods will still see the added bars, the evaluation does ignore the bars. The second method retains the original resolution but loses a part of the image, which will affect the prediction of all edges.

For this research, we have chosen to use the first method, because most of the images we use were in the same shape as the target shape, so they only had to be scaled without any black bars necessary. Furthermore, it significantly lessened the laboriousness of the process of finding suitable images for the categories, as the part of the image lost with the second method did not need to be taken into consideration.

3.4 Evaluation Metrics

Normally some of the metrics are unique to evaluating on a specific training and testing dataset. However, we chose to use all eight metrics used across the three methods as our data differs from any of the previous testing datasets. In addition to these eight metrics, we added a new one.

- **Accuracy threshold:** δ_1, δ_2, and δ_3 are used to quantify how many predicted pixels are within the threshold of 1.25 to the power of respectively 1, 2, or 3 when compared to the ground truth.

- **Local shape threshold:** ν_5, ν_4, and ν_3 are a new metric proposed by this thesis with three different thresholds. This metric takes the values of a 5 by 5 window in the predictions normalised by its center and compares this to the same window in the ground truth normalised by its center. It then checks whether the absolute difference is less than the threshold. The thresholds are $0.25^{\{5,4,3\}}$. The objective of this metric is to gain insight into the local shape differences between the predictions and ground truths, which can help determine whether the methods perform well on the local shapes.

$$\frac{\sum_{w=1}^{n}(\sum_{p=1}^{m} \nu_p < thresh)}{\sum_{w=1}^{n} w_m}$$

$$\nu_p = |w_p - \hat{w}_p|$$

  Where n is the amount of windows, which may overlap, in the image, and m is the amount of valid pixels in window w, the center also is an invalid pixel as it is always 0. w_p is the center normalised pixel in window w of the ground truth and $\hat{w}_p$ is a center normalised pixel in window $\hat{w}$ of a prediction.

- **Root Mean Square Error:** RMSE is used to get the standard deviation. This metric heavily punishes larger errors.

$$\sqrt{\frac{1}{n}\sum_{p=1}^{n}(d_p - \hat{d}_p)^2}$$

- **Root Mean Square Log Error:** RMSE log is similar to the RMSE, but uses the log-transformed prediction and actual value. This means the relative deviation is computed instead of the absolute deviation, which means the scale is irrelevant and it punishes outliers heavier when they are large relative to the ground truth.

$$\sqrt{\frac{1}{n}\sum_{p=1}^{n}(\log_{10} d_p - \log_{10} \hat{d}_p)^2}$$

- **Log$_{10}$ Error:** Log$_{10}$ gives the error of the pixels on a logarithmic scale.

$$\frac{1}{n}\sum_{p=1}^{n}|\log_{10} d_p - \log_{10} \hat{d}_p|$$

- **Absolute Relative Error:** AbsRel measures the error of the pixels in an image relative to the ground truth. This means that all values for AbsRel above 1 are over-estimations.

$$\frac{1}{n}\sum_{p=1}^{n}\frac{|d_p - \hat{d}_p|}{d_p}$$

- **Squared Relative Error:** SqRel is similar to the AbsRel, but squares the error instead of taking the absolute. This causes larger errors to be relatively more inflated than smaller errors, thus it penalises those larger errors more.

$$\frac{1}{n}\sum_{p=1}^{n}\frac{(d_p - \hat{d}_p)^2}{d_p}$$

d_p is a pixel in the ground truth and $\hat{d}_p$ is a pixel in the prediction. n is the amount of valid pixels in the image.

4 Results

In this section, we will present the data obtained from our experiments, which will be further discussed in Section 5.

4.1 Performance in General

In this subsection, we compare the categories based on method and training dataset in order to obtain insights into their general performances. The data obtained from running the experiments is summarised in Tables 2 to 7. While this representation of the data might be less conclusive about the strengths and weaknesses, as it might be too dependent on the training dataset, it is included to verify this and to better study the impact of the training. Before diving into the results per method and training, it is worth noting a few curiosities across the results, namely, the low performance of the outdoor categories on AbsRel and SqRel and the low performance of *Longer distance* overall, especially on the RMSE metric.

The content of the tables is sorted on δ_3 to make it easier to see the general performance per category. This is, however, not the definitive order from worst to best, as not all other metrics follow the same order, but the threshold is useful to determine what percentage of the pixels in the image is performing well, while the other metrics are influenced more by large outliers.

The results of the methods trained on NYU Depth V2 [9] are in Tables 2, 4, and 6. As this training dataset is completely comprised of indoor data, it is not unexpected to see the indoor categories perform generally better, with all methods having their best performance in *Close-ups* on nearly all metrics. However, *Longer distance*, while also an indoor category, is outperformed by all categories with BTS, and all categories except *Forest* with AdaBins and VNL. On the AbsRel metric, it is more on par with the other indoor categories. Furthermore, in Tables 2 and 6 the outdoor categories perform better on all metrics except for RMSE when there are fewer plants in the image. Yet in Table 4 *Forest* outperforms the other outdoor categories on SqRel as well. Lastly, we do see that, when sorting on δ_3, RMSE *log*, $\log_{10}$, and the other two thresholds are also roughly sorted from best to worst, although RMSE, AbsRel, SqRel, and local shape have an entirely different order.

When trained on KITTI [5] the results are obtained as seen in Tables 3, 5, and 7. This training dataset consists of outdoor images, which can explain the better performance of the three outdoor categories on most metrics. However, on AbsRel and SqRel the outdoor categories perform significantly worse, which can be explained by the larger depth range outdoor images often have. This can cause high AbsRel and SqRel with a low RMSE if the ground truth is close to the camera and the estimation is relatively far away from the camera. In this situation, the relative error measured by the former two metrics is higher than the absolute error measured by the latter metric. Remarkable is that *Longer distance* has a rather low performance with this training dataset as well, as one would assume the larger depth range would be more like the training data of KITTI. Instead, the methods perform worse in this category even when compared to the other categories, although the local shape metric does have better results than with some other categories. For example, *Forest* has a rather low performance on the local shape metric. As with the results in the other tables, the results obtained with KITTI as the training dataset are similar across categories as well. However, the metrics that follow δ_3 are slightly different. When trained on KITTI, RMSE almost perfectly follows the same order, and RMSE *log* has more differences. Log_{10} has roughly the same order with this training dataset as well.

Overall, when comparing the best and worst performing categories of the methods trained on KITTI to those trained on NYU Depth V2, the former are generally performing worse than the latter.

4.2 Performance Combined

In this subsection, we examine the outcome of combining the results based on the method, which might help gain insight into the performance of the methods independent of the training datasets. This is important as the training dataset influences the results, and we seek to determine properties of the methods, not the datasets. As mentioned in the previous subsection, the results follow a pattern that does not differ greatly per method and training dataset, which suggests that the influence of these datasets might be significant. In Tables 8, 9, and 10 these new results can be seen. While the general order is again similar, there are some curious differences.

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Forest	0.077	0.158	0.261	0.044	0.147	0.29	1.09	1.964	0.631	25.972	42.299
Longer distance	0.046	0.122	0.319	0.06	0.192	0.496	8.589	1.273	0.523	0.792	6.7
Parks	0.175	0.342	0.472	0.051	0.174	0.375	1.299	1.442	0.44	14.374	33.417
Man-made structures	0.223	0.424	0.604	0.119	0.382	0.692	1.099	0.971	0.308	6.234	15.343
Fake-light source	0.38	0.623	0.736	0.18	0.47	0.822	1.693	0.594	0.224	0.675	2.44
Plants indoors	0.369	0.692	0.836	0.116	0.36	0.672	2.249	0.578	0.2	0.647	2.146
Close-ups	0.248	0.634	0.883	0.241	0.558	0.885	0.76	0.456	0.172	0.434	0.472

Table 2: AdaBins [1] trained on NYU Depth V2, sorted on δ_3

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Longer distance	0.004	0.007	0.014	0.151	0.235	0.493	10.556	2.464	1.046	0.916	8.808
Plants indoors	0.032	0.09	0.199	0.169	0.391	0.687	3.158	1.281	0.53	0.726	2.456
Fake-light source	0.053	0.125	0.228	0.194	0.451	0.797	2.444	1.204	0.501	0.685	1.906
Close-ups	0.055	0.124	0.23	0.303	0.524	0.834	1.385	1.033	0.427	0.642	0.929
Parks	0.108	0.246	0.413	0.066	0.199	0.374	1.31	1.307	0.427	7.421	11.652
Man-made structures	0.164	0.358	0.519	0.183	0.482	0.717	1.077	0.94	0.328	2.636	3.051
Forest	0.198	0.411	0.581	0.061	0.159	0.278	0.612	1.565	0.436	12.516	10.165

Table 3: AdaBins [1] trained on KITTI, sorted on δ_3

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Longer distance	0.035	0.108	0.302	0.044	0.159	0.451	8.594	1.289	0.531	0.81	6.847
Forest	0.119	0.278	0.451	0.035	0.124	0.275	0.795	1.766	0.529	19.011	23.167
Parks	0.178	0.334	0.459	0.034	0.126	0.326	1.38	1.447	0.444	14.295	35.586
Man-made structures	0.192	0.386	0.558	0.068	0.246	0.59	1.32	1.05	0.342	7.652	26.645
Fake-light source	0.4	0.634	0.755	0.115	0.375	0.773	1.662	0.559	0.209	0.669	2.409
Plants indoors	0.347	0.642	0.812	0.084	0.286	0.613	2.302	0.603	0.212	0.717	2.329
Close-ups	0.27	0.665	0.904	0.138	0.432	0.825	0.731	0.44	0.163	0.43	0.494

Table 4: BTS [8] trained on NYU Depth V2, sorted on δ_3

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Longer distance	0.003	0.007	0.013	0.071	0.195	0.465	10.518	2.522	1.071	0.924	8.829
Plants indoors	0.032	0.075	0.152	0.112	0.339	0.658	3.227	1.412	0.582	0.765	2.649
Fake-light source	0.035	0.083	0.162	0.12	0.376	0.759	2.516	1.342	0.559	0.733	2.029
Close-ups	0.042	0.099	0.188	0.137	0.402	0.786	1.48	1.212	0.496	0.683	1.048
Parks	0.118	0.27	0.422	0.053	0.178	0.36	1.373	1.327	0.438	7.033	14.656
Man-made structures	0.171	0.326	0.472	0.123	0.395	0.685	1.158	1.024	0.362	2.751	4.523
Forest	0.24	0.452	0.602	0.046	0.15	0.282	0.666	1.494	0.412	11.21	13.581

Table 5: BTS [8] trained on KITTI, sorted on δ_3

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Forest	0.085	0.179	0.286	0.075	0.194	0.296	1.009	1.931	0.614	24.534	36.614
Longer distance	0.057	0.156	0.386	0.055	0.176	0.466	8.215	1.142	0.465	0.795	6.559
Parks	0.191	0.342	0.475	0.051	0.178	0.379	1.266	1.43	0.437	12.895	24.956
Man-made structures	0.203	0.392	0.564	0.095	0.324	0.646	1.239	1.027	0.332	7.022	20.465
Fake-light source	0.451	0.683	0.804	0.173	0.458	0.802	1.536	0.501	0.183	0.646	2.46
Plants indoors	0.373	0.683	0.83	0.122	0.356	0.658	2.218	0.575	0.199	0.672	2.081
Close-ups	0.396	0.795	0.945	0.202	0.514	0.853	0.648	0.375	0.133	0.411	0.514

Table 6: VNL [17] trained on NYU Depth V2, sorted on δ_3

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Longer distance	0.001	0.002	0.007	0.157	0.251	0.502	10.569	2.498	1.068	0.929	8.878
Plants indoors	0.012	0.038	0.104	0.197	0.432	0.726	3.238	1.414	0.594	0.763	2.554
Fake-light source	0.032	0.073	0.122	0.217	0.479	0.834	1.485	1.167	0.493	0.693	1.05
Close-ups	0.045	0.084	0.151	0.377	0.612	0.891	2.54	1.365	0.575	0.741	2.048
Parks	0.165	0.359	0.525	0.084	0.233	0.395	1.18	1.255	0.384	8.342	14.199
Man-made structures	0.24	0.432	0.581	0.211	0.521	0.733	1.015	0.914	0.308	2.868	4.196
Forest	0.245	0.469	0.625	0.063	0.172	0.291	0.59	1.57	0.424	13.64	11.964

Table 7: VNL [17] trained on KITTI, sorted on δ_3

AdaBins performs rather well in the *Man-made structures* category, which is approximately the best-performing category together with *Close-ups*. The other two outdoor categories, however, perform significantly worse on all but one metric. *Plants indoors* and *Fake-light source* follow *Close-ups* relatively close in terms of performance.

BTS performs well in *Forest*, especially on the RMSE and δ threshold metrics, but it underperforms somewhat on the other metrics. On the local shape metric it even performs worst out of all categories. *Close-ups* is performing rather nicely as well, especially on the AbsRel and SqRel, where it significantly outperforms most other categories. Of the outdoor categories, *Parks* has the worst performance, with a relatively large difference between this category and the other two categories. It is performing worse than the indoor categories, except for *Longer distance* on most metrics.

VNL performs well in *Man-made structures* and *Close-ups*, with *Parks* performing rather well as well. *Plants indoors*, while outperforming *Fake-light source* on δ_3, performs worse than the latter on all other metrics, which is not unique to VNL, but most noticeable for this method. *Forest* underperforms on most metrics with a relatively large margin.

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Longer distance	0.025	0.065	0.167	0.106	0.214	0.495	9.573	1.869	0.785	0.854	7.754
Forest	0.138	0.285	0.421	0.053	0.153	0.284	0.851	1.765	0.534	19.244	26.232
Parks	0.142	0.294	0.443	0.059	0.187	0.375	1.305	1.375	0.434	10.898	22.535
Fake-light source	0.217	0.374	0.482	0.187	0.461	0.810	2.069	0.899	0.363	0.680	2.173
Plants indoors	0.201	0.391	0.518	0.143	0.376	0.680	2.704	0.930	0.365	0.687	2.301
Close-ups	0.152	0.379	0.557	0.272	0.541	0.860	1.073	0.745	0.300	0.538	0.701
Man-made structures	0.194	0.391	0.562	0.151	0.432	0.705	1.088	0.956	0.318	4.435	9.197

Table 8: AdaBins [1] Tables 2 and 3 Combined, sorted on δ_3

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Longer distance	0.019	0.058	0.158	0.058	0.177	0.458	9.556	1.906	0.801	0.867	7.838
Parks	0.148	0.302	0.441	0.044	0.152	0.343	1.377	1.387	0.441	10.664	25.121
Fake-light source	0.218	0.359	0.459	0.118	0.376	0.766	2.089	0.951	0.384	0.701	2.219
Plants indoors	0.190	0.359	0.482	0.098	0.313	0.636	2.765	1.008	0.397	0.741	2.489
Man-made structures	0.182	0.356	0.515	0.096	0.321	0.638	1.239	1.037	0.352	5.202	15.584
Forest	0.180	0.365	0.527	0.041	0.137	0.279	0.731	1.630	0.471	15.111	18.374
Close-ups	0.156	0.382	0.546	0.138	0.417	0.806	1.106	0.826	0.330	0.557	0.771

Table 9: BTS [8] Tables 4 and 5 Combined, sorted on δ_3

Category	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
Longer distance	0.029	0.079	0.197	0.106	0.214	0.484	9.392	1.820	0.767	0.862	7.719
Forest	0.165	0.324	0.456	0.069	0.183	0.294	0.800	1.751	0.519	19.087	24.289
Fake-light source	0.242	0.378	0.463	0.195	0.469	0.818	1.511	0.834	0.338	0.670	1.755
Plants indoors	0.193	0.361	0.467	0.160	0.394	0.692	2.728	0.995	0.397	0.718	2.318
Parks	0.178	0.351	0.500	0.068	0.206	0.387	1.223	1.343	0.411	10.619	19.578
Close-ups	0.221	0.440	0.548	0.290	0.563	0.872	1.594	0.870	0.354	0.576	1.281
Man-made structures	0.222	0.412	0.573	0.153	0.423	0.690	1.127	0.971	0.320	4.945	12.331

Table 10: VNL [17] Tables 6 and 7 Combined, sorted on δ_3

NYU Depth V2

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	0.077	0.158	0.261	0.044	0.147	0.29	1.09	1.964	0.631	25.972	42.299
BTS	**0.119**	**0.278**	**0.451**	0.035	0.124	0.275	**0.795**	**1.766**	**0.529**	**19.011**	**23.167**
VNL	0.085	0.179	0.286	**0.075**	**0.194**	**0.296**	1.009	1.931	0.614	24.534	36.614

KITTI

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	0.198	0.411	0.581	0.061	0.159	0.278	0.612	1.565	0.436	12.516	**10.165**
BTS	0.24	0.452	0.602	0.046	0.15	0.282	0.666	**1.494**	**0.412**	**11.21**	13.581
VNL	**0.245**	**0.469**	**0.625**	**0.063**	**0.172**	**0.291**	**0.59**	1.57	0.424	13.64	11.964

Table 11: *Forest*. Bold is best, underlined is second best

4.3 Categories Compared Across Methods

In this subsection, we investigate the performance of the methods per category. For improving existing methods, it is not only necessary to look at the strengths and weaknesses of those methods, but also at how they perform compared to each other, as a strength of a method could still be worse than a weakness of another method, as an extreme example. To this end, we compare some of the categories on performance without combining the results of the two training datasets, as the compared values are all obtained using the same dataset. These tables do not have new results, but only a different presentation of the same results as in Section 4.1. We only mention and include tables of significant differences, not when the methods only differ slightly.

Table 11 shows that BTS performs relatively well in the *Forest* category when trained on NYU Depth V2 compared to the other two methods. When trained on KITTI, it is more tied with VNL. On the local shape metric, it performs approximately equally for both training datasets. In the category *Man-made structures* VNL performs better than the other two methods on most metrics when trained on KITTI as seen in Table 12, although AdaBins does not fall far behind. However, when trained on NYU Depth V2, AdaBins performs best and even achieves results comparable to the results of VNL trained on KITTI. In Table 13 we see that VNL outperforms the other methods on most metrics with both other methods performing similarly.

All indoor categories roughly follow the same pattern, where AdaBins performs best when trained on KITTI, VNL performs best when trained on NYU Depth V2, and BTS mostly performs second best in both cases, albeit with a similar performance compared to the third. The only metric that has contrary results is the local shape, where the performance of AdaBins and VNL is swapped. Notable as well is that AdaBins performs relatively well on SqRel in almost all experiments. *Close-ups* is a generally high-performing category, especially when trained on NYU Depth V2, but VNL also excels in this case, with only SqRel and the local shape thresholds not performing significantly better, as seen in Table 14. The other two methods are approximately even with BTS performing slightly better. In Table 15 the pattern previously mentioned is visible. For *Fake-light source* specifically, the other methods underperform significantly, while being at approximately the same level as each other. Table 16 shows that VNL performs well in *Longer distance* relative to the other methods.

NYU Depth V2

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	**0.223**	**0.424**	**0.604**	**0.119**	**0.382**	**0.692**	**1.099**	**0.971**	**0.308**	**6.234**	**15.343**
BTS	0.192	0.386	0.558	0.068	0.246	0.59	1.32	1.05	0.342	7.652	26.645
VNL	0.203	0.392	0.564	0.095	0.324	0.646	1.239	1.027	0.332	7.022	20.465

KITTI

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	0.164	0.358	0.519	0.183	0.482	0.717	1.077	0.94	0.328	**2.636**	**3.051**
BTS	0.171	0.326	0.472	0.123	0.395	0.685	1.158	1.024	0.362	2.751	4.523
VNL	**0.24**	**0.432**	**0.581**	**0.211**	**0.521**	**0.733**	**1.015**	**0.914**	**0.308**	2.868	4.196

Table 12: *Manmade structures*. Bold is best, underlined is second best

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	0.108	0.246	0.413	0.066	0.199	0.374	1.31	1.307	0.427	7.421	**11.652**
BTS	0.118	0.27	0.422	0.053	0.178	0.36	1.373	1.327	0.438	**7.033**	14.656
VNL	**0.165**	**0.359**	**0.525**	**0.084**	**0.233**	**0.395**	**1.18**	**1.255**	**0.384**	8.342	14.199

Table 13: *Parks*, trained on KITTI. Bold is best, underlined is second best

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	0.248	0.634	0.883	**0.241**	**0.558**	**0.885**	0.76	0.456	0.172	0.434	**0.472**
BTS	0.27	0.665	0.904	0.138	0.432	0.825	0.731	0.44	0.163	0.43	0.494
VNL	**0.396**	**0.795**	**0.945**	0.202	0.514	0.853	**0.648**	**0.375**	**0.133**	**0.411**	0.514

Table 14: *Close-ups*, trained on NYU Depth V2. Bold is best, underlined is second best

NYU Depth V2

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	0.38	0.623	0.736	**0.18**	**0.47**	**0.822**	1.693	0.594	0.224	0.675	2.44
BTS	0.4	0.634	0.755	0.115	0.375	0.773	1.662	0.559	0.209	0.669	**2.409**
VNL	**0.451**	**0.683**	**0.804**	0.173	0.458	0.802	**1.536**	**0.501**	**0.183**	**0.646**	2.46

KITTI

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	**0.053**	**0.125**	**0.228**	0.194	0.451	0.797	**2.444**	**1.204**	**0.501**	**0.685**	**1.906**
BTS	0.035	0.083	0.162	0.12	0.376	0.759	2.516	1.342	0.559	0.733	2.029
VNL	0.045	0.084	0.151	**0.217**	**0.479**	**0.834**	2.54	1.365	0.575	0.741	2.048

Table 15: *Fake-light source*. Bold is best, underlined is second best

Method	*higher is better*						*lower is better*				
	δ_1	δ_2	δ_3	ν_5	ν_4	ν_3	RMSE	RMSE *log*	$\log_{10}$	AbsRel	SqRel
AdaBins	<u>0.046</u>	<u>0.122</u>	<u>0.319</u>	**0.06**	**0.192**	**0.496**	<u>8.589</u>	<u>1.273</u>	<u>0.523</u>	**0.792**	<u>6.7</u>
BTS	0.035	0.108	0.302	0.044	0.159	0.451	8.594	1.289	0.531	0.81	6.847
VNL	**0.057**	**0.156**	**0.386**	<u>0.055</u>	<u>0.176</u>	<u>0.466</u>	**8.215**	**1.142**	**0.465**	<u>0.795</u>	**6.559**

Table 16: *Longer distance*, trained on NYU Depth V2. Bold is best, underlined is second best

5 Discussion

5.1 Strengths and Weaknesses of the Methods

With the results mentioned in Section 4, we can make assumptions about the strengths and weaknesses of the three methods. One general notion about the *Forest* category is that the images have a relatively high number of invalid pixels, while these pixels are not evaluated, they are mostly between branches where the depth value would be larger. If the methods fail to estimate the branches correctly and instead estimate them to be in the further away background, this will impact the AbsRel and SqRel metrics especially. This can explain the high results for those metrics in Table 11 and can be seen especially in the first row of Figure 8.

Longer distance. As this category is unique in its performance, even across the training datasets, we will mention our thoughts on it before analysing the data per method. *Longer distance* is performing significantly worse than the other categories. This could be attributed to the alienness of the category compared to the training datasets, as a large enclosed space is not widely present in those datasets. We do, however, see better performance within this category when looking at images with objects closer to the camera instead of all objects being further away. Notable is that even the images that perform the worst have a visual result that looks somewhat like the ground truth with the main difference being that the estimations place everything closer to the camera.

All in all, the category might underperform due to the training data lacking images with a relatively high minimum depth or due to the possibility that the methods might be unable to handle scenes without an object close to the camera as, for example, an anchor point from which to estimate the rest of the depth. The latter idea is further consolidated by looking at the local shape metric, as it has relatively high results for this category, suggesting the differences between the ground truth and the prediction are not necessarily large on a local scale, but perhaps only on a global scale. Some examples of the performance in *Longer distance* can be found in Figure 7. In this figure the first row performs relatively well on most metrics, the second row performs relatively well for only VNL, and the fifth row performs poorly on most metrics, especially on the δ threshold metrics. The last row falls in line with the notion that the category might only have worse performance due to its high minimum depth as the shape of the ceiling is visible, but the depth at which it is estimated differs.

Fake-light source and *Plants indoors* are special cases as well. In each of the methods, they perform adequately but not relatively good or bad, although due to their lower performance compared to *Close-ups*, they could be evaluated as a more neutral attribute of the methods and not a weakness or strength. One limitation of this evaluation is that both categories contain images that widely differ from other images in their respective category, which might affect the results negatively. However, as will be discussed in subsection 6.1, this is currently difficult to avoid.

5.1.1 AdaBins

AdaBins as a method should excel at tasks with a larger depth range as it can adapt to these ranges. However, it does not seem to perform better in the categories with complex depth. *Close-ups* has a good performance, as well as *Man-made structures* on most metrics. This might be due to the less complex scenes in these categories. The former has little difference in objects per image and the latter has mostly standard geometric shapes, albeit with some distortion due to perspective. These attributes can make it easier for AdaBins to find a linear combination of the bin centers without losing much detail.

While this combination of the bins seems to help with the former two categories, it is less clear with the more complex scenes like in the categories *Forest* and *Parks*. For example, in *Parks* AdaBins has a better visual result on the trees and shrubbery compared to the other methods, although the metrics do not support this. Figure 10 shows a few examples of this. However, compared to the other methods, most estimations made by AdaBins do seem to have the most detail. The low scores on the metrics could be explained by the estimation being off on the depth itself, while the relative depth across the image is correct.

Overall, supported by the tables, the strengths of AdaBins are its ability to achieve relatively good results when the training dataset does not contain scenes similar to the test dataset, and its performance on indoor scenes and scenes with mainly geometric shapes. Its weakness seems to lie in its ability to estimate images with large depth ranges when complex objects like plants are present.

5.1.2 BTS

BTS seems to perform well on outlines of objects, even in the *Forest* category it estimates the general outlines better than AdaBins and VNL. While this is mainly suggested by the metrics, in the images we see that BTS follows the general colour of most scenes better, as seen in Figure 8. This is most likely why it performs better on the metrics compared to the other two methods, as especially δ_3 punishes the output less when it is approximately right while still lacking detail. The local shape further supports this as its lower performance may indicate the details are not adequate. Interestingly, BTS performs relatively poorly in *Parks* compared to the other two outdoor categories. This might be due to the increased large changes in depth in this category, which does not combine well with the method BTS uses for its object outlining method. A scene with occlusion caused by small objects, like branches, especially has these object outlines poorly estimated.

In general, BTS achieves good results in specifically the *Forest* category, as well as scenes with almost no non-geometric shapes. Scenes with more complex occlusion due to, for example, branches of trees seem to perform worst, although in general, this method has the most constant performance across all categories with less pronounced strengths and weaknesses.

5.1.3 VNL

VNL tries to reconstruct a point cloud and with that information estimate the depth. This seems to be working quite well for most categories. In *Man-made structures* especially, the lines between concrete blocks are visible and in brick walls, the wall joints are often visible as well. These small details, while not evidently influencing the metrics, make the estimation look more correct. Some examples can be seen in Figure 9, especially in rows two and three. The presence of these geometric shapes in *Parks* might explain the performance of VNL in this category as well. This is supported by Figure 10, where it is noticeable that it can accurately estimate the structures, but lacks somewhat in estimating the plants.

On the indoor categories, the performance of VNL correlates with the previous notions. In Figure 6, especially in row four and five, and to a lesser degree in the other rows, we can see VNL performing worse compared to the other methods in estimating details of plants. This reinforces the idea that the method performs relatively poorly on plant-like objects. When trained on NYU Depth V2 it achieves results better than the other two methods. While at first it could be attributed to the training data, the differences with the other two methods imply that the method itself is in part responsible for the results.

On average, the strengths of VNL are, according to our data, estimating geometric shapes like buildings even with other shapes present, and especially correctly estimating details like lines between blocks or bricks. Complex scenes with plants, especially when not containing geometric shapes as well, are the weakness of this method.

5.2 Local Shape Differences

In this subsection, we will discuss the results of the local shape threshold and its implications. Notably, BTS performs worse than the other two methods in all categories for this metric, even the *Forest* category, in which it has the overall best performance. As this metric measures the differences between local shapes, it implies that those do not match often for the *Forest* category and the output is more like a blur which is approximately at the correct depth. This is supported by Figure 8, although the other methods do not perform significantly better on this metric while that would be suggested by the figures. Overall, this would imply that BTS would be better than the other methods in real scenarios within the *Forest* category, as accurately estimating the distance between the camera and the objects is arguably more important than the local shapes of the output being accurate.

Our assumptions about *Longer distance* are supported by the local shape threshold. While it does not perform on par with the other indoor categories, it significantly outperforms two out of three outdoor categories on this metric. This suggests that the current methods do not perform badly on local shapes. However, they seem to be having difficulty with the general layout of the scene, and cannot accurately estimate the distance between the objects and the camera, while being able to better estimate the depth differences within and between objects.

The last categories with interesting local shape results are *Fake-light source* and *Plants indoors*. As previously mentioned, across the other metrics, they perform generally equally. However, on this metric, *Fake-light source* performs significantly better. As with the other categories mentioned in this subsection, this implies that this category performs well on the local shapes compared to *Plants indoors*, which is not unexpected as the latter category has complicated shapes like plants which are more difficult to predict correctly in the local shape, while the general global shape is less difficult.

In general, the proposed metric is useful for determining whether a method is performing worse due to inaccurate estimations of local shapes or not. While this information is less useful to verify whether a method would perform well in real settings such as autonomous driving, it is a useful analysis metric which can be used to further improve methods.

5.3 Best and Worst Images

Apart from our categories, we look at the images individually to try to discern any pattern that might suggest a weakness or strength.

- As seen in Figure 4, VNL is rather accurate when estimating the corner of the wall and the ceiling, as well as the ceiling itself. The other two methods seem to often estimate the ceiling as an extension of the wall.
- The brighter light sources in *Fake-light source* have an aura of light around them in the images, which seems to affect the estimations. As seen in Figure 5 this is mostly the case, although row five shows an example where the methods still estimate the object well. While this can be mitigated by using better cameras as well, finding a method that can deal with this type of image can contribute to the accessibility of depth estimation software for private use.
- For *Forest* we see in Figure 8 that AdaBins estimates most details correctly, perhaps even better than BTS. Overall, all methods perform better when there is no clutter of branches, which can indicate they perform better in less complex scenes in this category.
- In *Man-made structures* the worst-performing images have a few traits in common. They are taken from a frog's-eye view, which causes the nearest point to be somewhat further away from the camera, and a large part of the image is occupied by the sky. While the sky itself is not evaluated, it might confuse the methods in this specific type of image, especially since KITTI does not contain images with a perspective pointed towards the sky. Figure 9 shows an example of this type of image.

6 Conclusions and Further Research

In this thesis, we sought to identify the strengths and weaknesses of AdaBins [1], BTS [8], and VNL [17]. In addition, we proposed a new local shape metric using center normalised windows.

We identified a generally poor performance of all three methods on images with no objects close to the camera, which might be caused by either the training datasets or the methods themselves. Moreover, we presented data supporting the conclusion that: AdaBins performs well in the *Close-ups* and *Man-made structures* categories while performing worse in the *Forest*, *Parks*, and *Longer distance* categories; BTS performs well in the *Close-ups* and *Forest* categories and worse in the *Parks* and *Longer distance* categories while maintaining the overall most constant performance of the three methods; VNL achieves good results in the *Close-ups* and *Man-made structures* categories while achieving worse results in the *Forest* and *Longer distance* categories, with the difference compared to AdaBins that VNL performs better in *Parks*. Lastly, we described specific scenes that achieve poor results, which we speculate is due to a considerable amount of clutter in the scene or a large part of the scene being occupied by the sky.

The new metric we propose, the local shape threshold ν, gives useful insights, especially for verifying whether a category performs well or not due to mismatched local shapes or different minimum depths compared to the ground truth. In particular, it was interesting to discover that in some cases the performance of the methods was the opposite on this metric compared to the existing metrics. This metric can be useful for further research as it can help to identify these weaknesses and strengths more easily. For example, our data implies that BTS has the most possible improvement in the local shapes, which can be a focus for further research in this specific method while learning from methods that perform better on this metric.

6.1 Further Research

In order to advance this research or other monocular depth estimation research, the obvious solution is to produce more, and more diverse, datasets. Not only for training, but also for testing can this be an improvement. One of the main struggles with finding strengths and weaknesses is that the data should be as similar as possible to each other and differ only on one or a few categories in order to more easily identify these qualities of the methods. For example, by taking a picture of the same scene with either different lighting, added and removed objects, or even the same objects with a different placement. With new specialised datasets, this research could be performed in a more reliable manner. Without producing new datasets, there could be research performing tests on different subcategories from the categories compared in this thesis, or even on more categories. In particular, research on the exact reason why images with no objects close to the camera achieve the results that they do in *Longer distance* would contribute to better understanding monocular depth estimation.

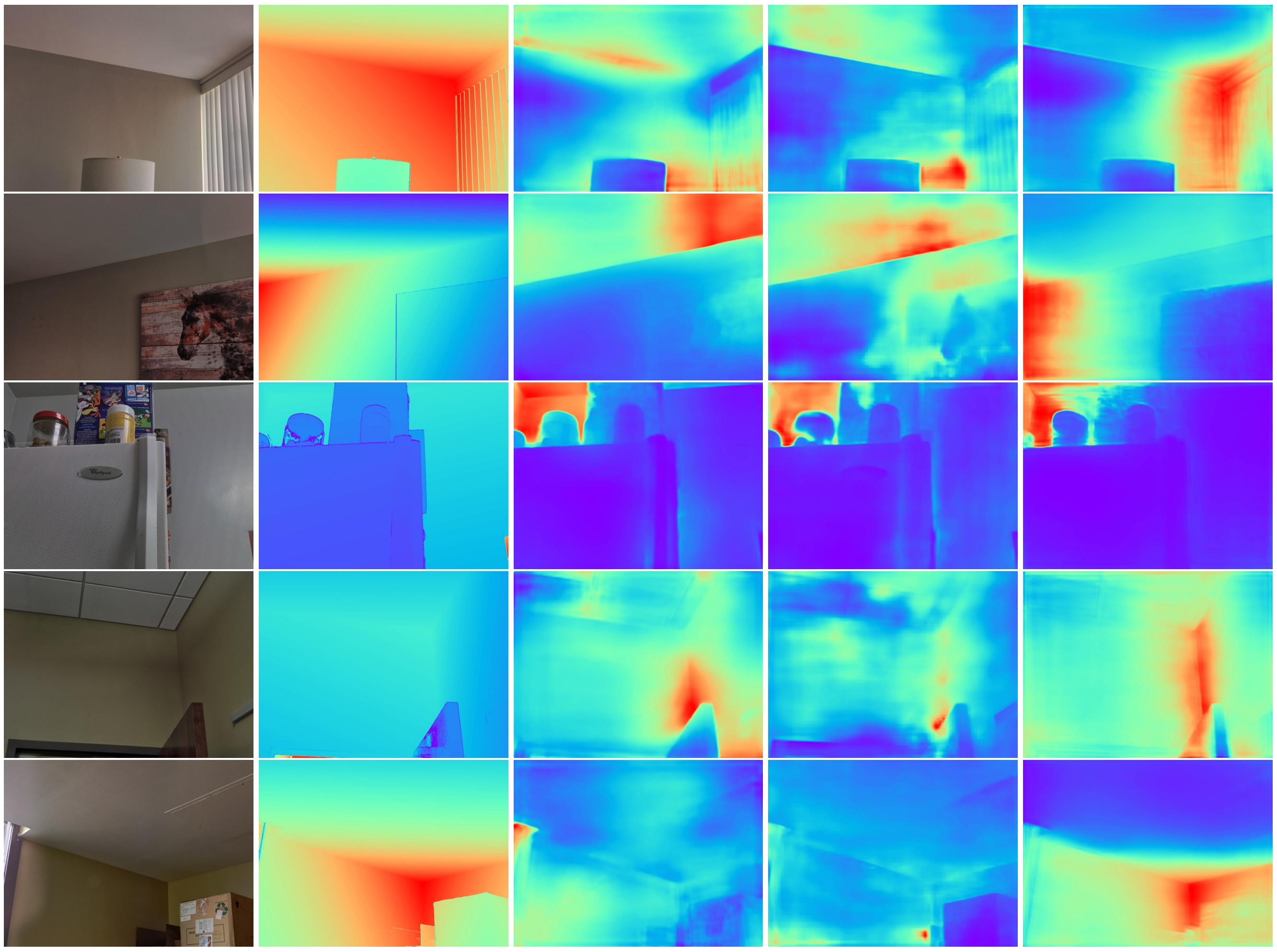

Figure 4: *Close-ups* trained on NYU Depth V2

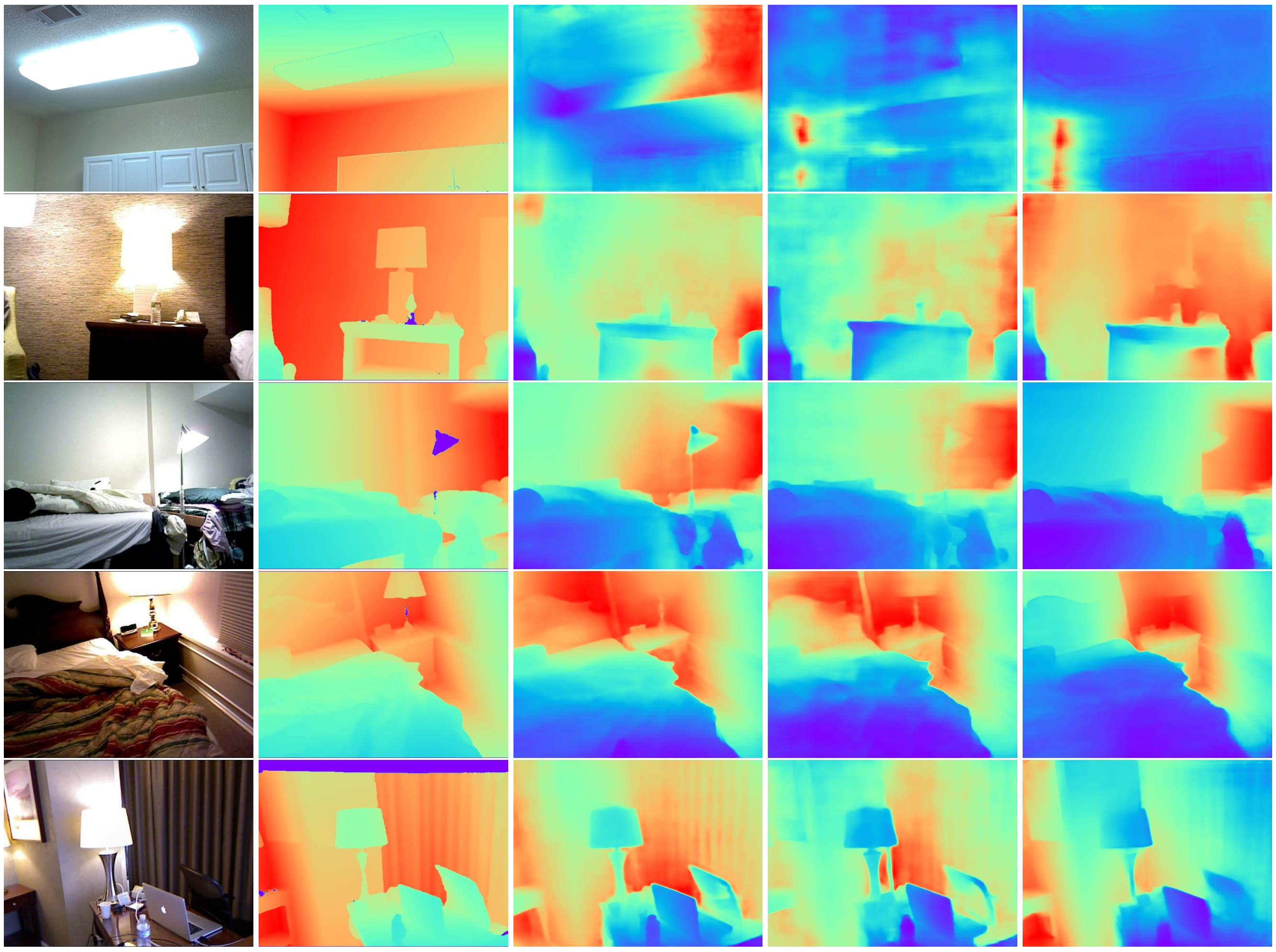

Figure 5: *Fake-light source* trained on NYU Depth V2

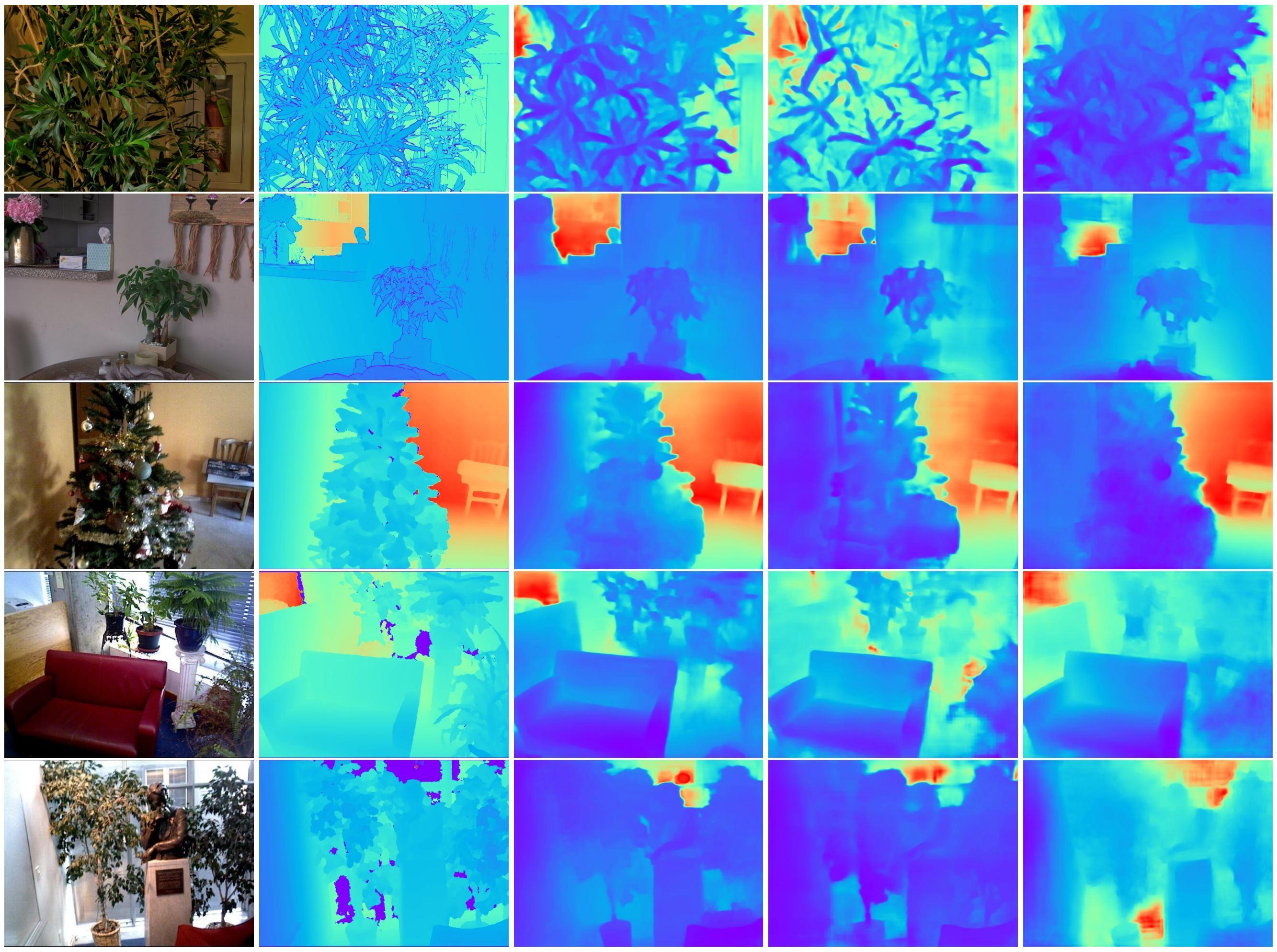

Figure 6: *Plants indoors* trained on NYU Depth V2

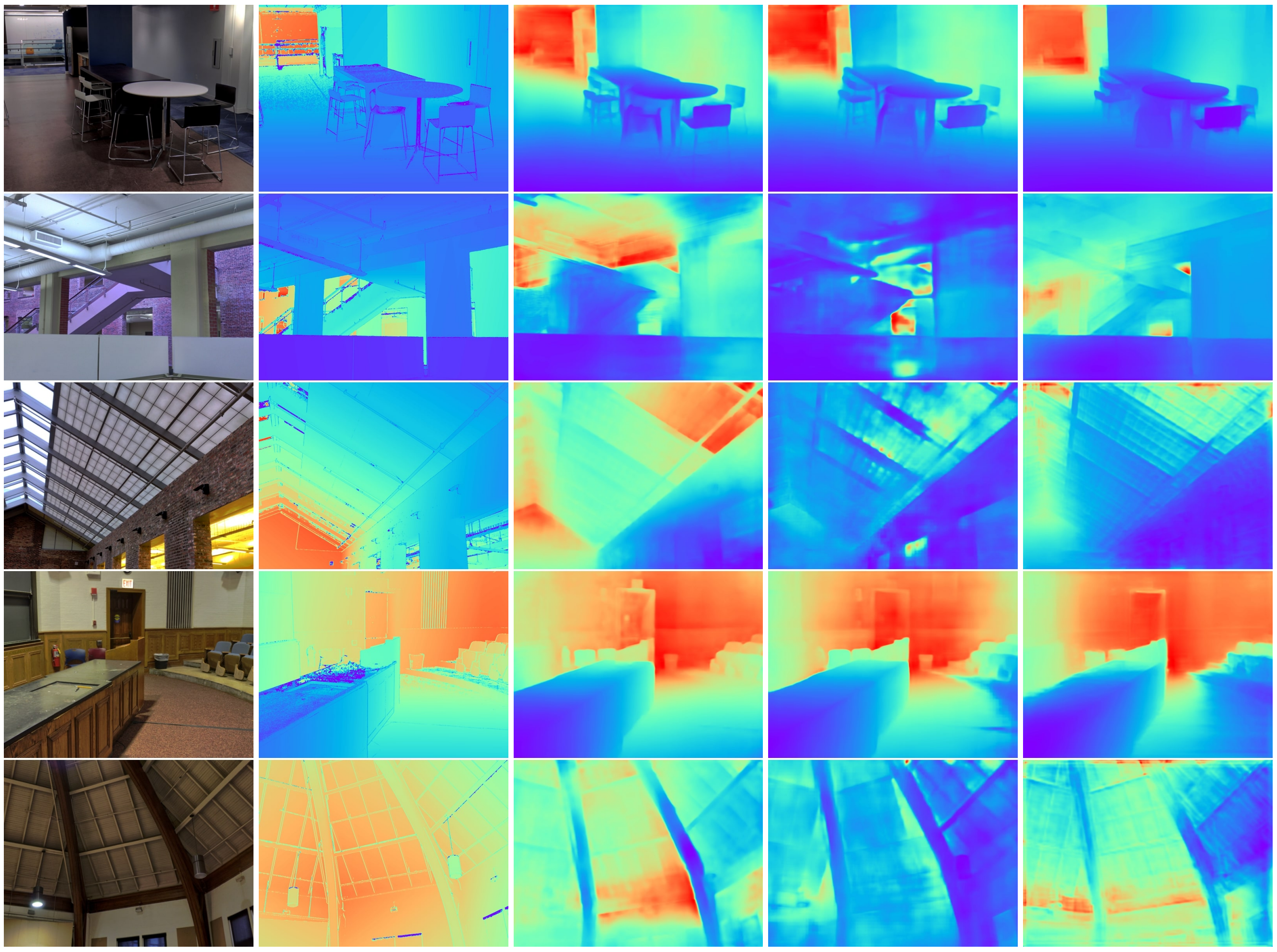

Figure 7: *Longer distance* trained on NYU Depth V2

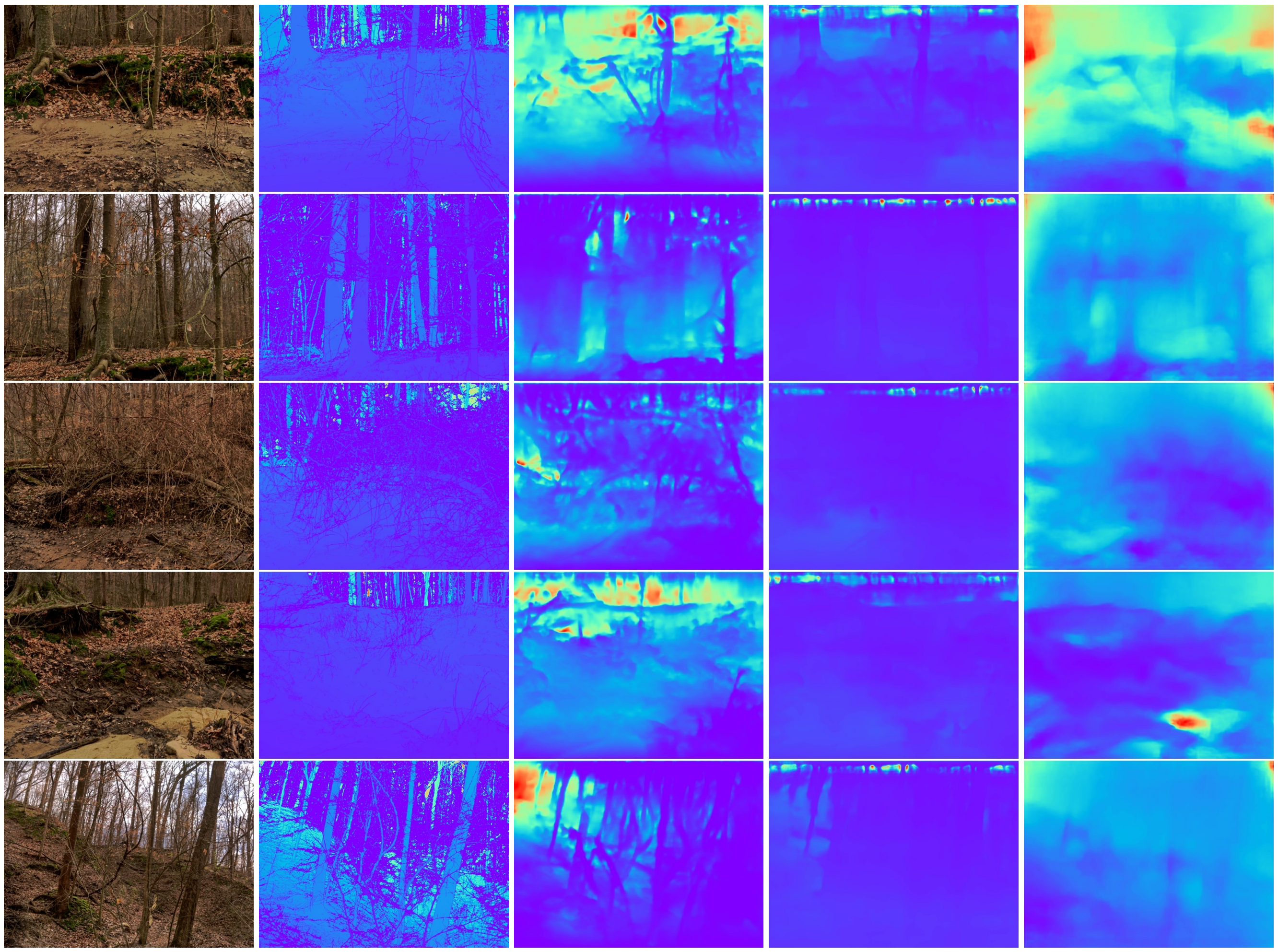

Figure 8: *Forest* trained on KITTI

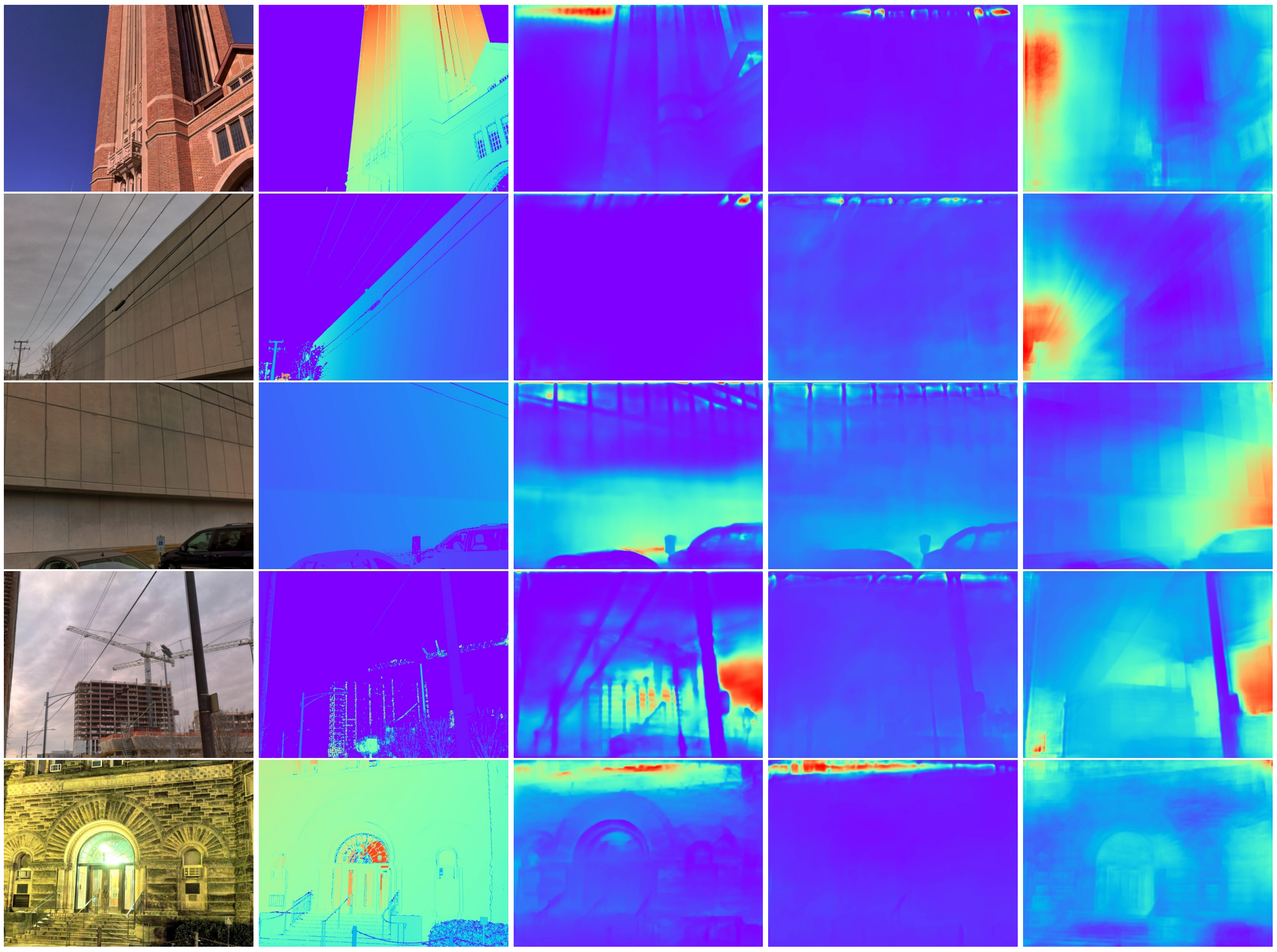

Figure 9: *Man-made structures* trained on KITTI

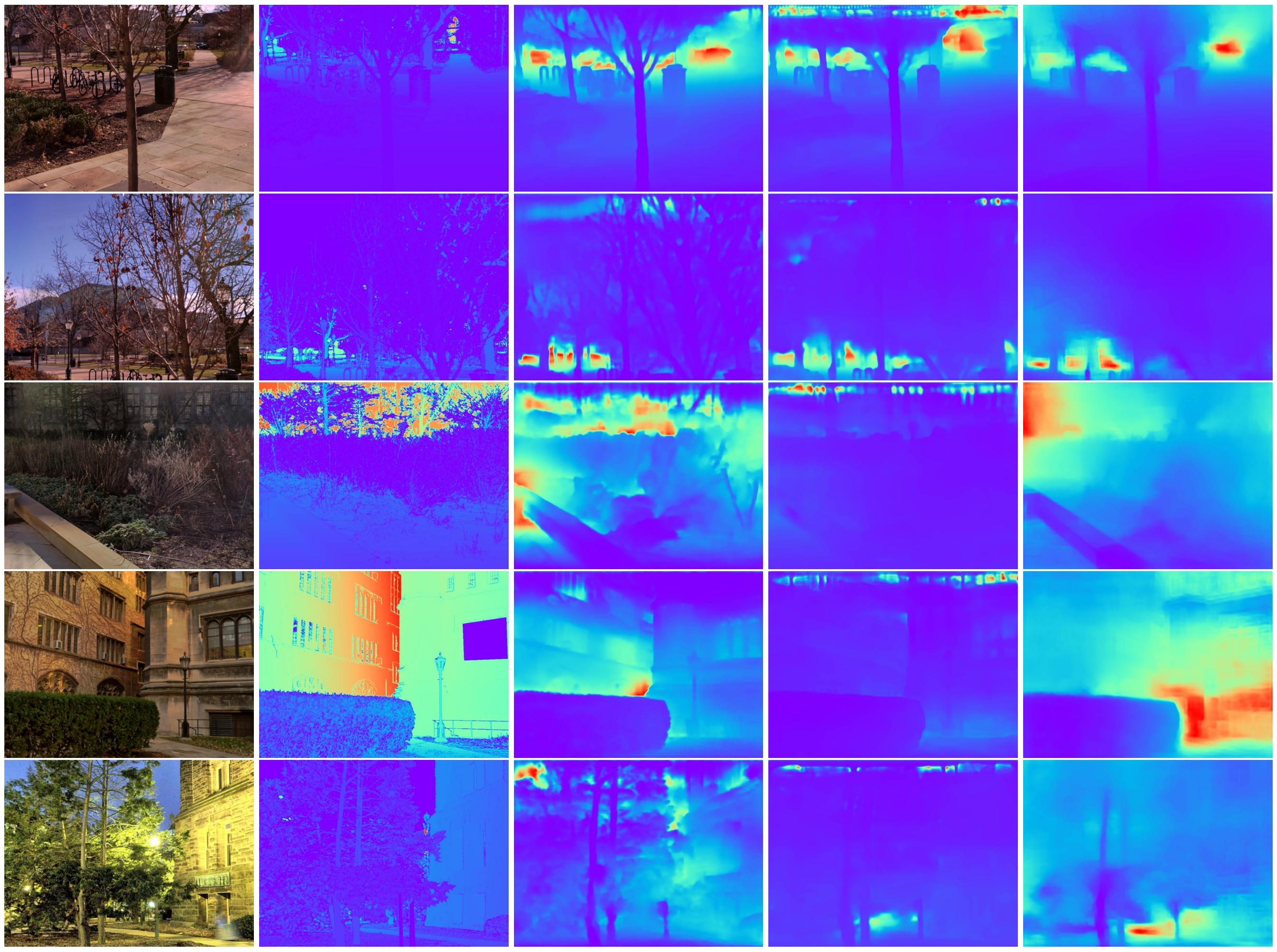

Figure 10: *Parks* trained on KITTI

References

[1] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021.

[2] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182, 2017.

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[4] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011, 2018.

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[6] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[7] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*, pages 141–165, 2013.

[8] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[9] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[10] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[11] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[12] Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. DIODE: A Dense Indoor and Outdoor DEpth Dataset. *CoRR*, abs/1908.00463, 2019.

[13] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE international conference on computer vision*, pages 1625–1632, 2013.

[14] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[15] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3684–3692, 2018.

[16] Wei Yin, Yifan Liu, and Chunhua Shen. Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[17] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[18] Wei Yin, Xinlong Wang, Chunhua Shen, Yifan Liu, Zhi Tian, Songcen Xu, Changming Sun, and Dou Renyin. Diversedepth: Affine-invariant depth prediction using diverse data. *arXiv preprint arXiv:2002.00569*, 2020.