Mapillary Planet-Scale Depth Dataset

Manuel López Antequera[1], Pau Gargallo[1], Markus Hofinger[2], Samuel Rota Bulò[1], Yubin Kuang[1], and Peter Kontschieder[1]

[1] Facebook
[2] Institute of Computer Graphics and Vision. Graz University of Technology

Abstract. Learning-based methods produce remarkable results on single image depth tasks when trained on well-established benchmarks, however, there is a large gap from these benchmarks to real-world performance that is usually obscured by the common practice of fine-tuning on the target dataset. We introduce a new depth dataset that is an order of magnitude larger than previous datasets, but more importantly, contains an unprecedented gamut of locations, camera models and scene types while offering metric depth (not just up-to-scale). Additionally, we investigate the problem of training single image depth networks using images captured with many different cameras, validating an existing approach and proposing a simpler alternative. With our contributions we achieve excellent results on challenging benchmarks before fine-tuning, and set the state of the art on the popular KITTI dataset after fine-tuning.
The dataset is available at mapillary.com/dataset/depth

1 Introduction

The availability of large-scale training datasets has significantly contributed to the rise of deep learning based approaches in computer vision. Starting with ImageNet [6] for image classification, also the quality of object detection [8,3] or semantic-, instance- and panoptic segmentation algorithms [5,18,22,31] has been greatly improved within a few years only. Yet, metric-accurate, large-scale, natural image datasets are still to come for the task of monocular depth estimation, most likely because they cannot be collected with commodity hardware in a straightforward way. Research in monocular depth estimation therefore predominantly use smaller, less varied or up-to-scale datasets for training [11,17,30]. Unsupervised methods are achieving remarkable results [13,12], but their performance still lags behind that of supervised methods.

For validation of single image depth methods, an important benchmark is the Make3D [24] dataset, comprising laser scans coupled with RGB images. Although dated, it is still a reference benchmark in the field. Recently, modern hardware has been used in a similar fashion to produce very high-quality datasets to be used as benchmarks for single-image depth methods such as DIODE [29] and iBims-1 [15]. Please refer to Tab. 1 for an overview of depth datasets.

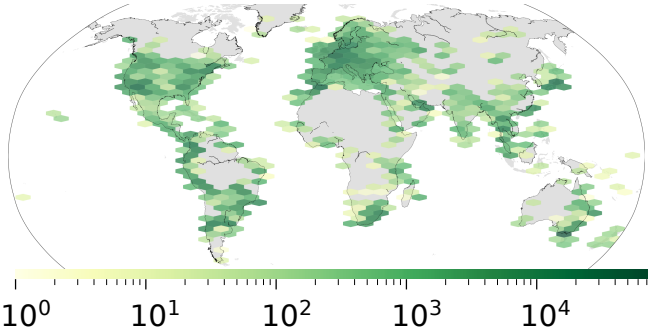


Fig. 1: Global distribution of the Mapillary Planet-Scale Depth (MPSD) dataset

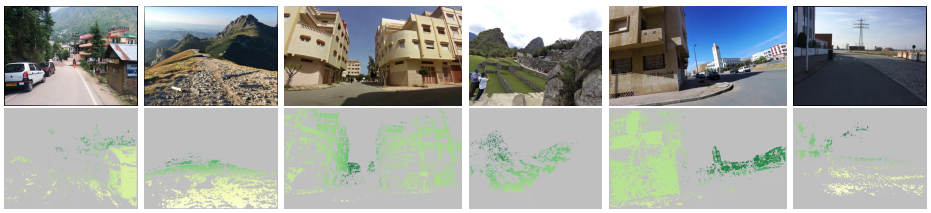


Fig. 2: Sample images and depth values from the proposed MPSD dataset

In this paper we introduce a novel, virtually arbitrarily scalable dataset – MPSD – providing training data for monocular depth estimation. Our dataset is solely derived from Mapillary's publicly available image database[3].

The dataset is generated by running monocular Structure-from-Motion and multi-view stereo we obtain dense depth for eligible images. Our dataset, containing images from all over the world, is larger, more complex and diverse than any previously published depth dataset. It currently comprises $\approx$ 750,000 images, extracted from over $50,000$ individual 3D reconstructions captured by a broad range of camera types with different focal lengths (Fig. 4) and distortion characteristics, in a broad set of environments (Fig. 1) and weather conditions, seasons, times of day, viewpoint and with real noise and motion patterns.

Training with such a dataset is not straightforward, as it is necessary to account for the heterogeneous cameras used to capture the images. This is a problem often overlooked and only recently studied by Fácil et al [9], where they demonstrate the advantage of explicitly accounting for the camera intrinsics during training. We successfully use their proposed CAM-Convs to train using our dataset, and also suggest an alternative, simpler technique to deal with the problem of multi-camera training.

[3] Currently holding $\approx 10^9$ images and corresponding GPS positions.

Table 1: Overview of depth datasets The proposed Mapillary Planet-Scale Depth (MPSD) dataset is large and diverse enough to effectively transfer onto several target datasets without fine-tuning. Refer to Section 4 for more details.

Dataset	n. Images	Source	Extent	Metric
Make3D [24]	534	Lidar	Palo Alto	yes
iBims-1 [15]	100	Lidar	Various scenes	yes
DIODE [29]	26 k	Lidar	25 Scenes	yes
KITTI [11]	94 k	Lidar	Karlsruhe	yes
WSVD [30]	1.5M	Stereo	7k videos	no
Cityscapes [5]	25 k	Stereo	50 Cities	yes
MegaDepth [17]	130 k	SfM	200 Scenes	no
MPSD	750 k	SfM	50k Scenes(Fig. 1)	yes

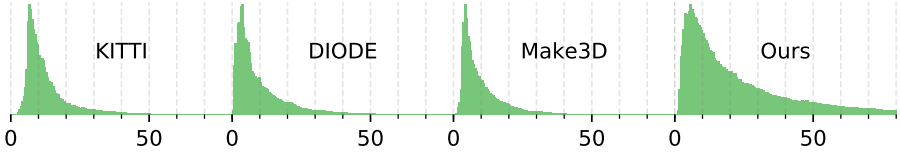


Fig. 3: Volume-normalized depth (m) distributions on several datasets

At the core of our work, we discuss the challenges to be tackled during the generation of MPSD from *real-world* data and how to take advantage of it in modern deep-learning based algorithms. We particularly address how to:

- Generate a *metric-accurate*-depth dataset from images captured in sub-optimal conditions for structure-from-motion such as low framerate, non-orbital trajectories, and under-constrained camera parameters.
- Effectively train deep neural networks for monocular depth estimation with data from many heterogeneous camera sources.

We conducted exhaustive ablations for the task of monocular depth estimation, proving the superior quality of our dataset against reference benchmarks like KITTI [11], MegaDepth [17], Cityscapes [5], DIODE [29] or Make3D [24]. With our approach and dataset, we achieve new state-of-the-art results for monocular depth prediction on the well-known KITTI benchmark.

2 Dataset

The Mapillary Planet-Scale Depth (MPSD) dataset contains 750,000 images and associated depth maps. It is based on imagery collected from Mapillary[4], on which we perform monocular structure-from-motion (SfM) to obtain relative

[4] Mapillary is a street-level imagery platform hosting images collected by members of their community.

camera poses. A multi-view stereo algorithm is then used to compute dense depth. Absolute (metric) depth is recovered from the GPS metadata that is available alongside the images. Although similar approaches to produce depth have been used on phototourism-style [27,17] datasets sourced from photography websites such as Flickr, Mapillary imagery poses new challenges when used in this manner.

Mapillary collects street-level imagery that is uploaded by individual users and organizations. It is a very heterogeneous source, presenting imagery captured in a wide range of conditions and locations with a vast number of cameras, both consumer-grade and professional. This type of imagery is of great interest as the community progresses towards algorithms that are expected to perform beyond small benchmarks. However, recovering depth from Mapillary images cannot be performed by using out-of-the-box SfM pipelines, as it presents some challenges not present in phototourism datasets: All images in Mapillary are uploaded as-is from thousands of different uncalibrated cameras, requiring self-calibration from the SfM pipeline itself. However, most sequences are not valid to perform self-calibration because they are captured using forward-facing cameras and under forward / zooming motion, underconstraining the camera parameters [28]. Moreover, capture is usually performed at a low framerate, increasing the baseline between consecutive frames, which makes the correspondence problem non-trivial.

Due to these sub-optimal conditions for perfoming structure-from-motion, we found no turnkey solution (including the MegaDepth [17] pipeline) that could extract valid depth at scale from Mapillary sequences. Our process is described in the following sections as three stages: 2.1) Global model-wise camera calibration, 2.2) Image search and 2.3) Reconstruction and multi-view stereo.

2.1 Global Model-wise Camera Calibration

Camera calibration is required for metric 3D reconstruction. Cameras are usually calibrated using a physical printed calibration pattern imaged under a variety of poses with respect to the camera. In the case of Mapillary imagery, this is not possible as there are thousands of independent users and camera calibration is not enforced on them by the Mapillary app used to record images. However, it is also possible to automatically obtain good calibration parameters with monocular SfM for a camera or set of cameras if there are enough images capturing a scene with a layout such that the camera parameters are constrained. This method for automatically calibrating cameras is common in the 'phototourism' scenario where a large number of images capture the same object, generally following orbital-like trajectories.We attempt to obtain camera parameters from Mapillary images in a similar fashion, however, the coverage offered by Mapillary is not optimal for this. Imagery is most often recorded from forward-facing cameras mounted on vehicles driven on roads. Motion is thus mostly linear and without rotation. Naively downloading imagery and attempting to perform reconstructions did not yield stable camera parameters. Instead, we sample sequences that are: 1. Dense enough (less than 5 meters and 30 degrees[5] between consecutive frames) in order

[5] We obtain an initial estimate of the turning angle as the angle between consecutive segments on the GPS track of the sequence.

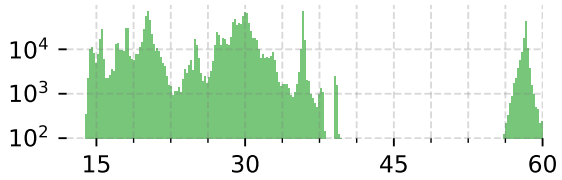


Fig. 4: Distribution of focal lengths (mm) in MPSD

to get enough point correspondences. 2. Have enough rotation (cumulative turn of 70+ degrees) in order to better constrain the focal length.

For each sequence, we compute the optimal calibration by running SfM reconstructions iteratively. We first run an incremental SfM algorithm with a default set of intrinsic parameters—focal length equiv. to 30mm and no radial distortion. The focal length and the first two distortion parameters of the Brown model are optimized during bundle adjustment. The result is a 3D reconstruction and an updated set of camera parameters. These parameters are used as initialization for a new SfM reconstruction, which in turn yields updated camera parameters. We iterate this process several times until the camera parameters stabilize. It is necessary to run the reconstruction process multiple times because improved initial camera parameters can improve the matching step which leads to better tracks and more constraints for the camera parameters.

Since the images at Mapillary are gathered by thousands of users with different devices, we can't obtain camera parameters for all of them as it isn't always the case that we can find adequate imagery on which to perform the aforementioned calibration process. We simplify the problem by assuming that all cameras reporting the same make, model, resolution and focal length will share the calibration parameters. In other words, we ignore differences due to manufacturing tolerances, temperature and so on. This simplification undoubtedly introduces some errors, but it is fundamental to make use of the imagery available in Mapillary, as there is rarely enough coverage from a single user to perform calibration on each user's camera independently.

To ensure that we have not calibrated the camera using an outlier (that is, a device whose calibration parameters deviate substantially from the modal parameters for that camera make and model), we run the calibration process for 10 different sequences for each camera make and model, and visualize the resulting camera parameters. We then manually confirm that calibrations from different sequences yield similar results and select one of them as the valid calibration for all images taken with that make and model, resolution and focal length.[6] Through this process we obtained calibrations for 250 camera models.

[6] Many action cameras and phones are able to capture under different 'modes' with different optics. We use the combination of reported focal length and resolution to disambiguate these modes.

2.2 Image Search

After calibrating a large set of camera models, we then mine Mapillary for images taken using these cameras and use them to perform SfM reconstructions and multi-view stereo to obtain depth. We start by selecting sampling weights w_r from 6 regions: North America: 20%, South America: 15%, Europe: 20%, Asia: 20%, Oceania: 15%, Africa: 10%. Each region is then partitioned into countries, assigning to each country a weight $w_c = w_\mathrm{r} \frac{\#\text{ims country}}{\#\text{ims region}}$

Each country is then partitioned in a regular grid of 156 by 156 km cells and the budget of images for that country is evenly distributed on this grid. We sample images randomly within each cell that: 1. Have been taken with one of the 250 cameras in our calibrated camera set. 2. Have at least 20 neighboring images in a radius of 10 meters (measured by the images' GPS tags). Each image and its neighbors are then used to perform a 3D reconstruction and multi-view stereo as described in the following section.

Further checks (described below) are performed after reconstructing in order to accept or reject this group of images into the dataset. If not accepted, another image from the cell is sampled. Since many cells are empty (rural areas or nature), we exhaust those and oversample more densely covered cells to satisfy the number of images allocated for that country. This is done to ensure that all of the images in underrepresented areas are used, obtaining as much diversity as possible.

2.3 Reconstruction and Multi-view Stereo

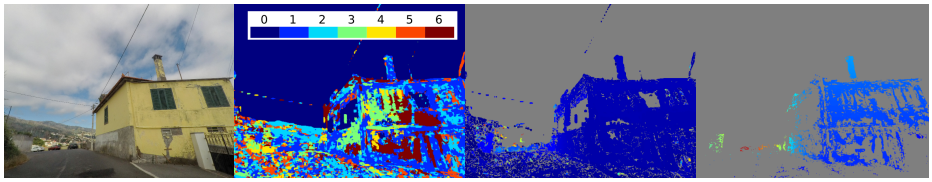
We perform structure from motion to obtain reconstructions of each of the candidate groups of images downloaded from Mapillary as described above. The reconstructions are performed using the OpenSfM [1] library with its default configuration settings. We chose OpenSfM due to prior familiarity, but other software [25,21] could be used to obtain similar results with appropriate settings.

We use the semantic segmentations available for each image in Mapillary to mask out regions that can negatively affect the reconstruction (pedestrians, vehicles, ego-vehicle and sky). After reconstructing, we obtain a set of sparse correspondences as well as relative camera poses.

The reconstruction is aligned to the GPS data associated with each image during bundle adjustment by adding a cost proportional to the squared distance between the GPS position and the reconstructed camera position. This fixes the scale ambiguity and yields metric distances, however, GPS measurements have noise that can affect this scale. In order to reduce the effect of the GPS measurement noise on the metric accuracy of the data, we filter out reconstructions that span a small region: After reconstructing[7], we check that the furthest two images are at least 20 meters apart, otherwise we discard the reconstruction.

In the experimental section of this paper we confirm that training on MPSD does indeed produce networks able to recover metric depth from single images.

[7] This check is performed only after reconstructing since it must be performed on images that can be registered to the reconstruction (some of the images in the neighborhood might have failed to reconstruct).



(a) Input image (b) Covisibility (c) PatchMatch (d) Clean depth

Fig. 5: Initial depth as obtained by multi-view stereo using PatchMatch (c) may contain spurious values. We clean the estimated depth by checking for consistency across several neighboring images (b). Only the depth values that are consistent with at least 3 neighbors are kept (d)

Using the relative poses obtained from SfM, we run a Patch-Match based multi-view stereo algorithm [26] to obtain dense depth estimates. This is a simple winner-takes-all stereo algorithm. Different depth and normal values are tested for each pixel and the one that gives the best normalized cross correlation score with the neighboring views is kept. The result is a dense but noisy depth map.

Most of the noise in the depth maps is removed in a post-processing step that checks the consistency between the depth maps of neighboring images. Depth values that are not consistent with at least 2 neighboring views are removed. This reduces the number of pixels for which a depth value is produced. We do not add any smoothness term to produce smoother depth maps nor do we try to inpaint the missing depth values.

The result is thus a set of 'clean' depth values that might be sparse, but that is reliable as shown in Figure 5. Finally, we discard depth maps in which less than 5% of the pixels have a depth value.

Opposite to what is done in the SfM step, during depth map estimation we do not mask out dynamic objects. The rationale is that we do want to have depth values on dynamic objects for training. While some dynamic objects are moving during capture and can possibly lead to wrong depth values, there are also many static-during-capture dynamic objects for which depth estimation will work. Additionally, when objects move in different directions than the camera, their motion does not satisfy the epipolar constraint and are easily rejected by the MVS algorithm. A notable exception are objects moving along the same road as the camera for which MVS produces scaled depth values. Manual analysis of our dataset finds very few examples of this, and training on MPSD produces networks that can predict depth on moving objects such as cars, a fact that we experimentally determine in Section 4.

3 Training with Multiple Cameras

The relationship between real world dimensions and pixels on the image plane for undistorted images as defined by the pinhole model is simple: The depth z of

an object is expressed in terms of its size in pixels the image plane y', its real size in meters y and the focal length of the camera f in pixels: $z = f\frac{y}{y'}$

Since we are dealing with the prediction of per-pixel depth values, we can simplify this expression for a single pixel ($y' = 1$): $z = fy$. Although single image depth is usually described as an ill-posed problem, it is solvable if y and f are known. It can be decomposed into two smaller problems: 1. Finding the focal length of the camera f, 2. Recalling the real dimensions depicted by pixel y'.

Most single-image depth methods deal with a single camera, simplifying the task. Learned models will implicitly memorize the value of the focal length f. This might be sufficient for applications using a single camera, as long as training data gathered with the same device is available. However, methods trained with a single camera will not generalize well to images captured by other devices.

Naively training on a dataset containing multiple cameras negatively impacts performance [9]. We hypothesize that this is because the network must accurately predict the focal length, a difficult task to perform, even when directly supervised to do so [14,20]. Focal length normalization alleviates this problem: The network is trained to predict $y = z/f$, a magnitude that only depends on the real world size of the area represented by the pixel of interest. This is quite effective and it has an intuitive explanation: the real-world size of objects is highly coupled with semantic segmentation, a task that convolutional neural networks excel at. To obtain a metric depth value during deployment, the predictions are multiplied by the focal length.

CAM-Convs [9] explicitly encode camera intrinsics by concatenating a map of the viewing directions in polar coordinates to each skip-connection in a u-net architecture. CAM-Convs are more general than just applying focal length normalization (although the authors found that it is beneficial to use both techniques in combination as it accelerates convergence), as it can also model different sensor sizes and aspect ratios explicitly. Images from different sensor sizes and resolutions can be resized and even *squashed* if necessary to fit the aspect ratio of the batch, as the network is explicitly informed about this through the appended features. It also adequately models the location of the principal point, enabling training on non-central crops.

In this work we experimentally validate CAM-Convs as a viable option to train single-image depth networks in datasets containing images taken from multiple cameras, while proposing a simpler alternative.

Camera Normalization. We suggest an alternative approach: resizing the images to approximate them being taken by a canonical camera with square pixels, focal length f_c and no radial distortion. With camera normalization, the relationship between the pixel size and the real size for any given object depends only on the depth, simplifying the task of depth prediction. For example, if the canonical focal length is $f_c = 700$px, an object of height $y = 2$m will have depth that is inversely proportional to its size in the image $z = 700 * 2/y'$ pixels. By resizing the input images to always have the same focal length, the network only needs

to learn to regress the real-world sizes of objects, as the focal length prediction isn't required anymore.

Instead of *informing* the network about the viewing angles as is done when using CAM-Convs, these angles are intrinsically learned by the model, as every input pixel always corresponds to the same viewing angles during training. The network does not need to produce different responses for similarly-looking patches as is the case when using CAM-Convs. Moreover, by resizing images to a fixed focal length, the range of pixel sizes at which objects are represented is reduced. For example, if the focal length is variable, the smallest objects will look even smaller on images with small focal lengths, and the larger objects would look even larger on images with large focal lengths.

Unlike CAM-Convs (that require a u-net like architecture), this approach is independent of the architecture, as it depends only on scaling the input images by a factor of f_c/f. In other ways, our technique is less flexible than CAM-Convs: Images are cropped or padded to a common size to form batches during training, and non-central crops can't be used. However, we didn't find these drawbacks to be of practical relevance on our experiments.

4 Experiments

Architecture We use a single architecture for all of our experiments, except in those cases where we compare against the implementations offered by other authors. We do so in order to offer a fair comparison and to focus on the differences in datasets and techniques for handling training with several cameras. The network is an encoder-decoder with skip connections (u-net) architecture, with a dilated resnet-50 pre-trained on ImageNet as the encoder. The dilation rates are 1,1,2 and 4 for each of the four residual modules, producing a feature map 16 times smaller than the input image. We use in-place activated batch normalization [2] to reduce the memory footprint during training allowing for large batches. Input size is always 1216 x 352 pixels, regardless of the scaling and cropping strategy.

After the encoder we append a DeeplabV3 [4] head to aggregate contextual information. It is formed by a set of dilated convolutions with different dilation rates (12, 24 and 36) as well as a global pooling of the features whose outputs are concatenated, batch normalized and convolved together to form an output feature map of the same dimensionality as the input.

This feature map is then upsampled through bilinear interpolation in three stages, each stage doubling the resolution. Features from the matching level in the encoder are concatenated to the upsampled features before being fed to a 'skip module' consisting of a convolution and activation. When using CAM-Convs, the viewing angles and normalized camera coordinates (8 channels) are resized to the corresponding shape and concatenated to the upsampled features and used as input to the skip modules. The final output of the u-net is thus at half of the resolution of the input image. We upsample once more to fit the size of the ground truth before computing the loss or evaluating.

Datasets We train on MegaDepth as a baseline and compare with our proposed MPSD dataset. When training our own models[8], we only use the subset of MegaDepth ($\approx$ 100k out of $\approx$ 130k images) that contains euclidean depth, as the rest of the dataset only contains *ordinal* (foreground/background) depth relationships).

During training, we use the KITTI validation set to track performance and perform early stopping. We then evaluate in a range of datasets (without any fine-tuning): Make3D [24], DIODE [29](outdoor), Cityscapes [5], MegaDepth [17] and KITTI [11]. We follow the usual practice of filtering out depth values that are unreliably large, removing values larger than 80 meters for all datasets except MegaDepth (no metric depth) and Make3D (70 m).

Scaling and cropping strategies We experiment with both CAM-Convs and our proposed camera normalization. Since CAM-Convs allow training using non-central crops, we evaluate both central and random crops when training using CAM-Convs. As a baseline, we also train our architecture on MegaDepth without any explicit handling of the focal length on the input images or the architecture: The network is fed with undistorted images that are simply resized so that their width is 1216 and then center-cropped to a common size of $(1216, 352)$.

Training details We train the network to predict the logarithm of the focal length-normalized depth and minimize the loss proposed in [7]:

$$L(z, z^*, f) = \frac{1}{n}\sum_i d_i^2 - \frac{\lambda}{n^2}\left(\sum_i d_i\right)^2 \tag{1}$$

where $d_i = \log(z) - \log(z^*)$. The loss is only evaluated on those pixels with known depth, where n is the number of valid depth points in the image. When training on MegaDepth, we use the fully scale-invariant version with $\lambda = 1$. We note that using a scale-invariant component in the loss leads to faster convergence, even if the training data is metric depth, thus, when training on MPSD, we set $\lambda = 0.5$.

We use stochastic gradient descent with an initial learning rate of 0.015, Nesterov momentum of 0.9 and weight decay of 10^{-4} for a maximum of 200k steps, stopping early if the performance on the KITTI validation set decreases. The learning rate is decayed on every step following $\mathrm{lr}_i = 0.015(1 - i/200,000)^{0.2}$. The batch size is set to 64, distributed over 8 V-100 GPUs.

MPSD vs. MegaDepth Training using our dataset (rows 6-9 in Table 2) yields better performance across the board when compared to MegaDepth, with the exception of evaluating on MegaDepth itself (row 5). Although MPSD does not exclusively contain driving scenarios, the type of imagery available in our dataset is mostly street-level imagery similar to KITTI and Cityscapes, which could be an explanation for the large gap. However, networks trained on our dataset also generalize well to Make3D and DIODE which are not datasets captured in

[8] We also compare with the model offered by the authors, trained on their full dataset.

Table 2: Results obtained by training on MegaDepth or MPSD. MD-Ordinal is the model trained by the authors of MegaDepth using their full dataset (including ordinal data, supervised with their ordinal loss). All other entries share the architecture from Section 4 and are trained with euclidean depth. "mini MPSD" is the MPSD dataset reduced to the size of the euclidean subset of MegaDepth. Scaling strategies are *Naive*: Resize and center crop to a fixed size, *CC*: CAM-Convs, *FF*: camera normalization. Crop strategies are (R)andom and (C)enter. We only report RMSE on methods trained with MPSD, as the scale is arbitrary when trained on MegaDepth. The best result is highlighted in bold. Entries fine-tuned on the target data are marked with *. In those cases, the second-best is also highlighted with bold text.

		Strategy		KITTI		MegaDepth	Cityscapes		DIODE (outdoor)		Make3D	
#	Training set	Scale	Crop	SILog	rmse	SILog	SILog	rmse	SILog	rmse	SILog	rmse
1	MD-Ordinal	-	-	30.1	-	10.8	35.19	-	47.52	-	38.2	-
2	MegaDepth	Naive	C	25.61	-	11.86	65.11	-	42.91	-	59.89	-
3	MegaDepth	CC	R	26.92	-	10.67	62.92	-	50.3	-	54.24	-
4	MegaDepth	CC	C	23.79	-	11.51	60.08	-	47.28	-	55.9	-
5	MegaDepth	FF	C	26.79	-	**9.96***	36.73	-	48.28	-	41.64	-
6	mini MPSD	FF	C	14.89	4.87	17.85	22.61	9.05	44.43	8.44	29.55	5.99
7	MPSD	FF	C	**12.77**	4.21	14.68	19.77	**7.91**	42.2	7.78	**27.49**	**5.54**
8	MPSD	CC	C	13.33	**4.13**	21.5	34.83	12.77	43.04	8.05	54.66	59.45
9	MPSD	FF+C	C	12.8	4.39	**14.04**	**19.52**	8.13	**41.69**	**7.75**	28.07	5.67
10	MPSD+KITTI	FF	C	**9.23***	**3.04***	32.23	27.11	8.58	45.55	10.69	37.56	6.49

driving scenarios. The size and variety in MPSD allows networks to generalize much better to all of the datasets we tried on.

It's not only size that matters We carry out an experiment to ascertain if the size of the MPSD dataset is crucial for the performance gains obtained when using it as a training set instead of MegaDepth. To do so, we factor out the size difference between MPSD and MegaDepth. We randomly sample our dataset to reduce it to the same size as the MegaDepth euclidean depth subset (around 100,000 images) and train on it. The results can be found on the sixth entry in Table 2 'mini MPSD': Performance on all the validation sets is only slightly worse than using the full dataset, while still much better than networks train on MegaDepth. We hypothesize that this is because the domain of MegaDepth is quite limited: although it is not a small dataset, the images in it display a small set of monuments and landmarks (the images were reconstructed into 200 distinct reconstructions). In contrast, the images from our dataset have been gathered from more than $50,000$ independent reconstructions all over the globe.

Metric accuracy of MPSD Although we obtain state of the art results when training on MPSD for the scale-dependant RMSE metric, we perform a simple experiment to determine if there is scale bias in MPSD: Using a network trained exclusively on MPSD, we produce depth predictions on several metric depth datasets and compute a scale factor as a least squares solution to align each

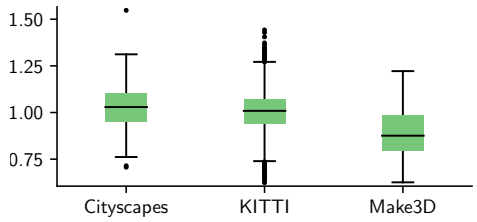


Fig. 6: Scale factors to align predictions of a MPSD-trained model (Entry #7 in Table 2) to the ground truth depth. Although there is some fixed scale that clearly improves results on Make3D, it is not the case for KITTI or Cityscapes, indicating that the depth in MPSD is indeed metric.

predicted depth frame with the ground truth depth. The resulting scale factors are aggregated on Figure 6, with average scales of 1.03, 1.01, and 0.89 for Cityscapes, KITTI and Make3D. The fact that we find a consistent underestimation of the depth on only one of the datasets implies that there is no scale bias in our dataset (networks trained on MPSD underestimate depth more often on Make3D due to the domain gap between MPSD and Make3D).

Scaling and cropping strategies Table 2 collects all of our experimental results for single image depth. We report the scale-invariant SILog [7] score in all cases. Since the MegaDepth dataset does not provide metric depth, we only report the root-mean-square error (in meters) for networks trained on MPSD.

We compare naively resizing the images versus using either CAM-Convs or camera normalization to train on MegaDepth. Our experiments confirm the observations of Fácil et al [9]: Accounting for the camera intrinsics explicitly greatly benefits training using datasets collected from more than one camera[9].

When comparing the two methods for dealing with multiple cameras, we found no clear winner. Both produce similar results, with the CAM-Convs producing the best results for some datasets and camera normalization in others. We also combined both methods (row #9, Table 2), resulting in the best performance for some of the evaluations. In this case, the CAM-Convs degenerate into a scaled version of Coord-Convs [19], a constant mapping of the viewing directions.

When using CAM-Convs, there is reason to believe that random crops may be used more effectively, as the concatenated viewing directions convey information about the cropped region. We experimented with using random crops during training (see rows #2 and #3 in Table 2) and found no conclusive results. We suspect that this is due to the wide aspect ratio for our crop size, as randomly cropping using this adspect ratio means in practice that the top and bottom of images will be sampled more often (usually containing regions with no ground truth depth like parts of the ego-vehicle or the sky).

[9] Refer to [9] for a thorough evaluation about the need of accounting for the focal length when training on datasets with multiple cameras.

Fig. 7: MPSD includes valid depth points on dynamic objects such that depth networks trained on MPSD are not "*dynamic-blind*". These examples are produced by a network trained exclusively on MPSD and evaluated on KITTI.

Table 3: RMSE(m) on the KITTI validation set, separated in to static (traffic signs etc.) and dynamic regions (pedestrians, cars, bicycles etc.) regions

Training set(s)	**Static**	**Dynamic**
MegaDepth	93.04	117.98
MPSD	**4.16**	**5.16**
MegaDepth+KITTI	3.74	4.29
MPSD+KITTI	**3.12**	**3.52**

Dynamic objects In Section 2.3 we described how we include depth from static-during-capture dynamic objects in MPSD. To demonstrate that the included depth is valid (that is, that a network trained on MPSD can recover depth on dynamic objects), we have devised a simple experiment. We trained two versions of our architecture, one on MegaDepth, and one on MPSD. Each version is then fine-tuned on KITTI, leading to four different architectures. We then compare all of them on the KITTI validation set.

For each image, we first run a state-of-the-art segmentation network [23] to separate each pixel into dynamic or static. We then run the depth prediction network on the image and align the depth prediction to the ground truth[10]. Finally, we calculate the RMSE for dynamic and static regions and gather the results in Table 3. The small gap between the dynamic and static regions when training on MPSD indicates the presence of quality annotations on the dynamic regions.

Competing in the KITTI public benchmark We have reported results after training on our diverse large scale dataset and evaluating on other smaller scale datasets that are well-known in the community without performing any fine-tuning.

[10] This is so that the MegaDepth-trained network can be included in this comparison.

Table 4: KITTI leaderboard at the time of submission. Simple supervised pre-training on MPSD can outperform methods trained with new techniques such as ordinal regression [10] and planar guidance [16].

Rank	Method	SILog	sqErrorRel	absErrorRel	iRMSE
1	MPSD	**11.12**	**2.07** %	8.99 %	**11.56**
2	GSM (Anon.)	11.23	2.13 %	8.88 %	12.65
3	GSM (Anon.)	11.56	2.25 %	8.99 %	12.44
4	LCI (Anon.)	11.63	2.20 %	9.07 %	12.42
5	BTS [16]	11.67	2.21 %	9.04 %	12.23
6	AcED (Anon.)	11.70	2.45 %	9.54 %	12.51
7	DORN [10]	11.77	2.23 %	**8.78** %	12.98

However, the best-known benchmark to date is still the KITTI depth dataset. The authors offer a test server to ensure fair comparison on a test set with held-out ground truth. Entries in this benchmark use KITTI and (optionally) other data to train their networks.

To compete in the benchmark, we fine-tuned the network trained on MPSD and our proposed camera normalization scaling method (entry #7 on Table 2) on the KITTI training set for 3 epochs, resulting in entry #10 on Table 2. We evaluated the held-out set using this network and submitted our predictions to the official benchmark server, obtaining a SILog score of 11.12, surpassing all other entries at the time of submission. However, it is worth noting that, after fine-tuning, the network performs worse on all the other benchmarks than a network trained only on MPSD. This is evidenced by comparing entries #7 and #10 from Table 2: We believe that future research should focus on the cross-dataset scenario.

5 Conclusion

We have presented the generation procedure of Mapillary Planet-Scale Depth (MPSD), a depth dataset automatically generated from geo-tagged RGB images. The dataset has an unprecedented scale, geographical span, variety in appearance, and range of capturing devices. Additionally, we have addressed the difficulties that arise when using a dataset taken by a large heterogeneous set of cameras when training single-image depth estimation, comparing the existing CAM-Convs with a proposed alternative.

MPSD is larger and more varied than any other publicly available depth dataset, obtaining state-of-the-art results on several benchmarks in the cross-dataset scenario, where no fine-tuning is allowed. We also achieves a new state of the art result on the KITTI single-image depth benchmark by using MPSD to pre-train a depth network that is then fine-tuned on the benchmark.

References

1. OpenSfM. https://github.com/mapillary/OpenSfM 6
2. Bulo, S.R., Porzi, L., Kontschieder, P.: In-place Activated BatchNorm for Memory-Optimized Training of DNNs. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition pp. 5639–5647 (2018). https://doi.org/10.1109/CVPR.2018.00591 9
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. CoRR **abs/1903.11027** (2019) 1
4. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking Atrous Convolution for Semantic Image Segmentation (2017), http://arxiv.org/abs/1706.05587 9
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes Dataset for Semantic Urban Scene Understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 1, 3, 10
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: Computer Vision and Pattern Recognition (CVPR) (2009) 1
7. Eigen, D., Puhrsch, C., Fergus, R.: Depth Map Prediction from a Single Image using a Multi-Scale Deep Network (2014), http://arxiv.org/abs/1406.2283 10, 12
8. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal visual object classes (VOC) challenge. Intern. Journal of Computer Vision (IJCV) **88**(2), 303–338 (2010) 1
9. Facil, J.M., Ummenhofer, B., Zhou, H., Montesano, L., Brox, T., Civera, J.: CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth (2019), http://arxiv.org/abs/1904.02028 2, 8, 12
10. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep Ordinal Regression Network for Monocular Depth Estimation (2018). https://doi.org/10.1109/CVPR.2018.00214, http://arxiv.org/abs/1806.02446 14
11. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3354–3361. IEEE, IEEE (jun 2012). https://doi.org/10.1109/CVPR.2012.6248074, http://ieeexplore.ieee.org/document/6248074/ 1, 3, 10
12. Godard, C., Mac Aodha, O., Brostow, G.: Digging Into Self-Supervised Monocular Depth Estimation (jun 2018), http://arxiv.org/abs/1806.01260 1
13. Gordon, A., Li, H., Jonschkowski, R., Angelova, A.: Depth from Videos in the Wild: Unsupervised Monocular Depth Learning from Unknown Cameras (2019), http://arxiv.org/abs/1904.04998 1
14. Hold-Geoffroy, Y., Sunkavalli, K., Eisenmann, J., Fisher, M., Gambaretto, E., Hadap, S., Lalonde, J.F.: A Perceptual Measure for Deep Single Image Camera Calibration. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (dec 2017). https://doi.org/arxiv:1712.01259, http://arxiv.org/abs/1712.01259 8
15. Koch, T., Liebel, L., Fraundorfer, F., Körner, M.: Evaluation of cnn-based single-image depth estimation methods. In: Leal-Taixé, L., Roth, S. (eds.) European Conference on Computer Vision Workshop (ECCV-WS). pp. 331–348. Springer International Publishing (2018) 1, 3

16. Lee, J.H., Han, M.K., Ko, D.W., Suh, I.H.: From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation (2019), http://arxiv.org/abs/1907.10326 14
17. Li, Z., Snavely, N.: MegaDepth: Learning Single-View Depth Prediction from Internet Photos (2018). https://doi.org/10.1109/CVPR.2018.00218, http://arxiv.org/abs/1804.00607 1, 3, 4, 10
18. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. CoRR **abs/1405.0312** (2014) 1
19. Liu, R., Lehman, J., Molino, P., Such, F.P., Frank, E., Sergeev, A., Yosinski, J.: An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution (NeurIPS), 1–26 (2018), http://arxiv.org/abs/1807.03247 12
20. López-Antequera, M., Marí, R., Gargallo, P., Kuang, Y., Gonzalez-Jimenez, J., Haro, G.: Deep Single Image Camera Calibration with Radial Distortion. In: Computer Vision and Pattern Recognition (CVPR) (2019) 8
21. Moulon, P., Monasse, P., Marlet, R., Others: Openmvg. https://github.com/openMVG/openMVG 6
22. Neuhold, G., Ollmann, T., Rota Bulò, S., Kontschieder, P.: The Mapillary Vistas dataset for semantic understanding of street scenes. In: Intern. Conf. on Computer Vision (ICCV) (2017) 1
23. Porzi, L., Rota Bulò, S., Colovic, A., Kontschieder, P.: Seamless scene segmentation. In: CVPR (2019) 13
24. Saxena, A., Sun, M., Ng, A.Y.: Make3D: Learning 3D Scene Structure from a Single Still Image. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(5), 824–840 (may 2009). https://doi.org/10.1109/TPAMI.2008.132 1, 3, 10
25. Schonberger, J.L., Frahm, J.M.: Structure-from-Motion Revisited. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4104–4113 (2016). https://doi.org/10.1109/CVPR.2016.445, http://ieeexplore.ieee.org/document/7780814/ 6
26. Shen, S.: Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. IEEE Transactions on Image Processing **22**(5), 1901–1914 (May 2013). https://doi.org/10.1109/TIP.2013.2237921 7
27. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring photo collections in 3D. In: SIGGRAPH Conference Proceedings. pp. 835–846. ACM Press, New York, NY, USA (2006) 4
28. Sturm, P.: Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (June 1997). https://doi.org/10.1109/CVPR.1997.609467 4
29. Vasiljevic, I., Kolkin, N., Zhang, S., Luo, R., Wang, H., Dai, F.Z., Daniele, A.F., Mostajabi, M., Basart, S., Walter, M.R., Shakhnarovich, G.: DIODE: A Dense Indoor and Outdoor DEpth Dataset (2019), http://arxiv.org/abs/1908.00463 1, 3, 10
30. Wang, C., Lucey, S., Perazzi, F., Wang, O.: Web stereo video supervision for depth prediction from dynamic scenes. CoRR **abs/1904.11112** (2019), http://arxiv.org/abs/1904.11112 1, 3
31. Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving video database with scalable annotation tooling. CoRR **abs/1805.04687** (2018) 1