IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 21, 2020, accepted December 30, 2020, date of publication January 11, 2021,
date of current version January 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050556

HighRes-MVSNet: A Fast Multi-View Stereo Network for Dense 3D Reconstruction From High-Resolution Images

RAFAEL WEILHARTER AND FRIEDRICH FRAUNDORFER, (Member, IEEE)
Institute of Computer Graphics and Vision, Graz University of Technology, 8010 Graz, Austria

Corresponding author: Rafael Weilharter (rafael.weilharter@icg.tugraz.at)

This work was supported in part by the European Institute of Innovation and Technology (EIT) RawMaterials under Project 18004, and in part by the RESilient transport InfraSTructure to extreme events (RESIST) Project through the European Union's Horizon 2020 Research and Innovation Program under Grant 769066.

ABSTRACT We propose an end-to-end deep learning architecture for 3D reconstruction from high-resolution images. While many approaches focus on improving reconstruction quality alone, we primarily focus on decreasing memory requirements in order to exploit the abundant information provided by modern high-resolution cameras. Towards this end, we present HighRes-MVSNet, a convolutional neural network with a pyramid encoder-decoder structure searching for depth correspondences incrementally over a coarse-to-fine hierarchy. The first stage of our network encodes the image features to a much smaller resolution in order to significantly reduce the memory requirements. Additionally, we limit the depth search range in every hierarchy level to the vicinity of the previous prediction. In this manner, we are able to produce highly accurate 3D models while only using a fraction of the GPU memory and runtime of previous methods. Although our method is aimed at much higher resolution images, we are still able to produce state-of-the-art results on the Tanks and Temples benchmark and achieve outstanding scores on the DTU benchmark.

INDEX TERMS Convolutional neural network, dense 3D reconstruction, multi-view stereo.

I. INTRODUCTION

Multi-View Stereo (MVS) attempts to reconstruct a highly detailed 3D model of an observed scene from images with different viewpoints. The prerequisites are known intrinsic and extrinsic camera parameters which can be obtained via Structure from Motion (SfM) (see Fig. 1). MVS has been a well studied problem for decades and traditional methods based on geometric context [2], [6], [7], [26] achieved great success when reconstructing scenes with Lambertian surfaces, especially in terms of *accuracy*. However, they struggle with the reconstruction of low-textured, specular, and reflective regions and in terms of *completeness*. Furthermore, they usually take a very long time to establish the 3D correspondence and larger scenes can take several hours to process.

To address these issues more recent approaches [12], [15] use deep Convolutional Neural Networks (CNNs) which are several times faster while also improving the overall 3D reconstruction quality of a scene. This can be mostly attributed to the fact that learning-based methods can incorporate global semantic information such as specular and reflective priors for more robust matching. Furthermore, if the

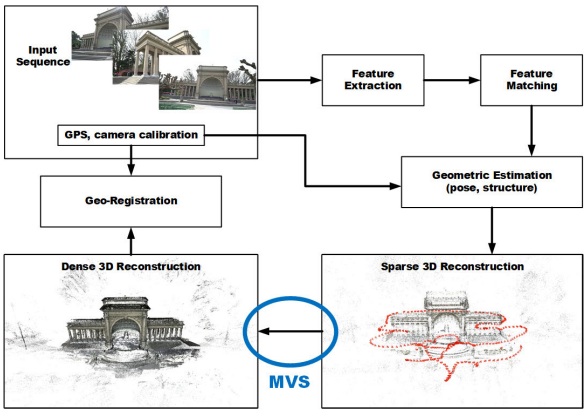


FIGURE 1. Overview of the Structure from Motion pipeline. MVS attempts to create a denser, more appealing 3d model from sparse reconstruction information.

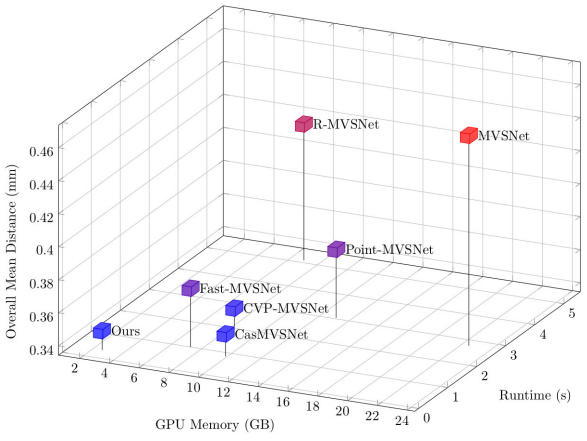
The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

receptive field of the CNN is large enough, poor textured regions can be better reconstructed.

Many networks [9], [29], [33] follow the approach of Yao *et al.* [32] and build a cost volume based on a plane sweep process and variance metric to estimate a depth map for every reference image. The cost volume is then regularized by applying multi scale 3D convolutions, which is an extremely memory intensive operation, growing cubically with the image resolution. This issue has been addressed by several subsequent works by either sequential regularization [33], reducing the depth dimension of the cost volume [9] or working with a sparse cost volume [35].

(a) Point Clouds: CasMVSNet (top) vs Ours (bottom)



(b) DTU benchmark performance

FIGURE 2. (a) Comparison between one of the best performing methods on the DTU dataset CasMVSNet (top) and Ours (bottom). (b) Although we only use a fraction of the GPU memory and runtime, we can still reconstruct small details in the 3D model.

However, even these latest MVS networks are already at the limit of current consumer grade GPU memories when using an input image size of only 2 Megapixels (see Fig. 2). With the unprecedented ubiquity of inexpensive high-resolution cameras (e.g. phones), we saw the need for an efficient network that is able to exploit this abundant information. The value of high-resolution data to increase accuracy has been explored for the binocular stereo case by Yang *et al.* in their HSM-Net [30]. They suggest to search for correspondences incrementally over a coarse-to-fine hierarchy and achieve impressive results on the respective stereo datasets [22], [24].

In this article, we propose HighRes-MVSNet which combines an hierarchical correspondence search through feature pyramid encoding with the cascading cost volume formulation. Our main contributions can be summarized as follows:

- We propose a novel MVS network architecture that utilizes insights from algorithms designed for the binocular stereo case to address the issue of 3d reconstruction from high-resolution images.
- We achieve state-of-the-art results on challenging benchmark datasets while significantly reducing GPU memory requirements and runtime. In particular, our method is at least 8× faster while requiring 6× less memory when compared to other MVS methods using the same input image resolution.
- To the best of our knowledge, this is the first MVS network that can effortlessly process input images of 12MP on a consumer grade GPU (e.g. NVIDIA GeForce GTX 1080 Ti).

The remainder of this article is organized as follows: In Section II, we present an overview of related works and their achievements. Subsequently, we explain our method in detail in Section III. Implementation details about parameter choices can be found in Section IV. We evaluate our method on well known datasets and conduct experiments in Section V. Finally, we conclude our findings in Section VI.

II. RELATED WORK

Traditional MVS methods use handcrafted features and the projection relationship between multiple views to optimize the depth value of each pixel. An example is Colmap by Schoenbergerand and Frahm [25], Schoenberger *et al.* [26] which performs well in a multitude of scenarios ranging from public benchmarks to internet photo collections. However, one of the major downsides of such traditional methods is their long processing time. The classic patch-match approach can take several minutes to estimate a depth map for a single image.

An alternative to traditional handcrafted features is to use learned features. In recent years, learned features have reached an unprecedented performance in image detection, segmentation and classification tasks [5], [8], [11], [19], [23]. Pioneered by Han *et al.* [10], the learning approach has also been applied to the two-view stereo case. Zbontar and LeCun [36] and Luo *et al.* [21] extract features with siamese networks and build a traditional cost volume followed by classic post-processing. Subsequent methods [3], [16], [17] apply 2D/3D convolutions to regularize the cost volume and thereby replacing the post-processing step. In particular, Yang *et al.* [30] propose an encoder-decoder architecture which searches for correspondences incrementally over a coarse-to-fine hierarchy. Such end-to-end learning algorithms remarkably boosted the performance and outperform traditional stereo approaches on the KITTI benchmark [22].

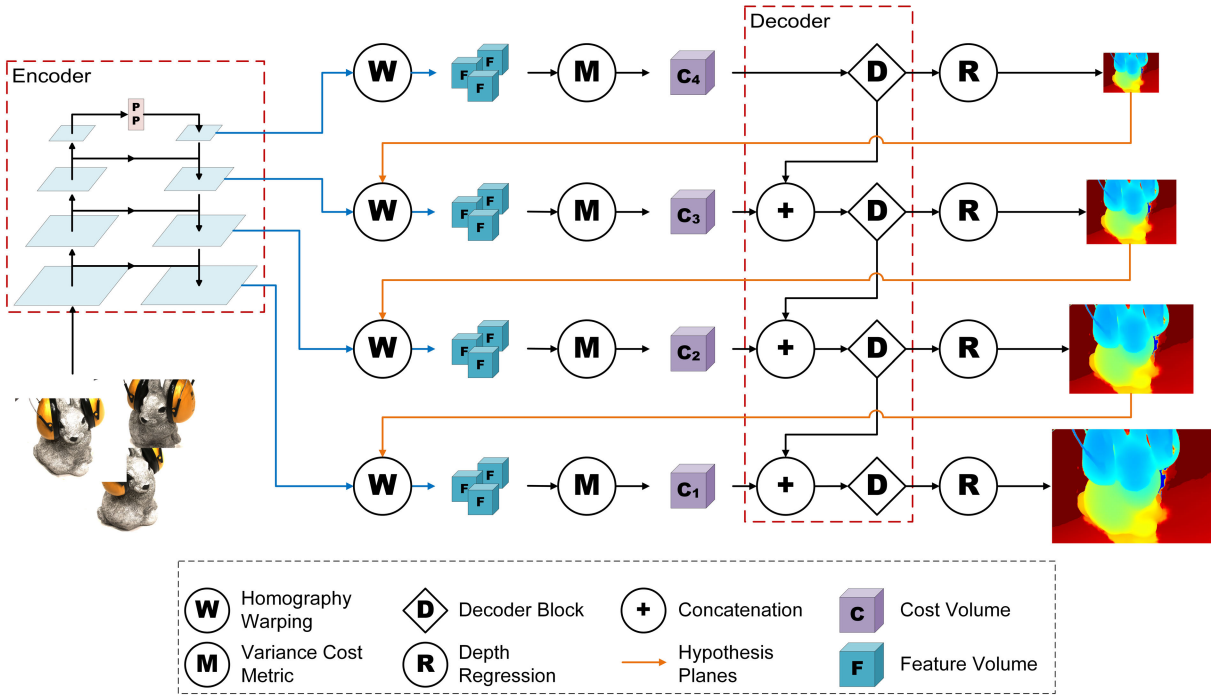


FIGURE 3. The proposed network architecture of HighRes-MVSNet. Given a set of images, features are extracted on 4 different scales in the encoder part utilizing pyramid pooling (PP). Then we assemble a cost volume at the coarsest scale (C_4) through differentiable homography warping and the variance cost metric. Next, the decoder produces 2 outputs: 1) A cost volume, which will be upsampled and fused with the raw cost volume of the next stage. 2) A classified cost volume, which through depth regression yields a depth map to initialize the feature volumes in the next stage. This process is repeated for 4 stages until we get our final output depth map.

Inspired by the success of learned features and cost volume regularization for the stereo case, MVS methods picked up on this idea. First approaches [14], [15] use a volumetric scene representation where the cost volume is built upon. A common drawback here is that due to memory requirements only small scale reconstructions are possible. Yao *et al.* [32] propose to compute the variance of cost volumes and produce a depth map for one reference image at a time. Depth maps can then be fused into a single point cloud. This allows for an adaptive reconstruction of a large scene.

However, even this one image at a time approach has severe memory issues when it comes to higher resolution images. To combat the high memory consumption of 3D cost volumes, several attempts have been made: R-MVSNet [33] tackles the problem by sequentially regularizing the 2D cost maps along the depth dimension via the gated recurrent unit (GRU). Nevertheless, this comes at the cost of efficiency and increases the runtime significantly. CasMVSNet [9] addresses the issue by building the cost volume upon a feature pyramid encoding geometry. Then the depth search range can be narrowed down by the prediction of the previous stage thus decreasing the depth dimension of the cost volume. In their Fast-MVSNet [35], Yu and Gao explore a sparse-to-dense approach: They only construct a sparse cost volume to learn a sparse, high-resolution depth map and densify local regions afterwards.

Even though the above mentioned networks achieve great results on public benchmark datasets [1], [18], they are easily brought to their limits when using input data with a resolution of over 2 Megapixels. Moreover, when it comes to the reconstruction of a larger dataset the seemingly small processing time will add up.

We combine several insights of these previous works to create HighRes-MVSNet, a network that is able to significantly reduce memory requirements and runtime while still achieving superior accuracy.

III. METHOD

We introduce HighRes-MVSNet, a deep CNN that is especially designed to handle high-resolution images as input. This is done by encoding features from the input image to different coarser scales. Afterwards, we utilize the cascading cost volume formulation [9] to predict the depth for every image in a coarse-to-fine manner. Figure 3 shows an overview of our architecture.

A. FEATURE EXTRACTION

We follow [30] in using an encoder-decoder architecture with skip connections and pyramid pooling (see Fig. 4). In the encoder, the network first applies 3 convolutions with stride 2 in the first layer, followed by a pooling layer and another convolutional layer with stride 2. This is done to already reduce the feature volume size to $\frac{1}{8}$ of the input while still taking into account all of the given information.

We then apply a unet architecture to extract features at 4 different scales: $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$ and $\frac{1}{64}$ as the coarsest scale

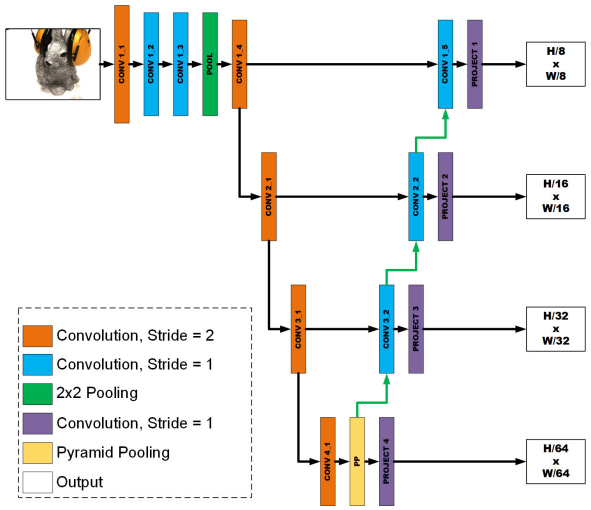


FIGURE 4. A detailed overview of our encoder architecture. The input is aggressively down sampled at the beginning through several convolutions and a pooling layer. A U-net architecture is then applied to extract features at different scales. The final convolution layer in each stage is used to project the features to a lower dimensional subspace to control the size of the output feature volume. An upscaling by a factor of 2 is indicated by the green arrows.

at level 4. At every scale, the features are projected to a lower dimensional subspace to control the size of the resulting feature map.

As in [30] our reasoning behind this approach is twofold: 1) Through this coarse-to-fine design principle, we drastically increase the receptive field. 2) Memory requirements are severely reduced by encoding the high-resolution image to $\frac{1}{8}$ of its original size at the first output stage.

This promising concept has successfully been used in two-view stereo, but to the best of our knowledge has not been extended to MVS.

B. COST VOLUME FORMULATION

As many MVS networks before [9], [29], [31], [33], we apply the differentiable homography warping operation [32] to build 3D cost volumes from the previously extracted feature maps at every scale. In its essence, this operation warps all feature maps into different fronto-parallel planes in the reference camera frustum. This process is similar to that of the classical plane sweeping stereo and the depth range is usually determined from the sparse reconstruction. The warping is defined by the homography:

$$H_i(d) = K_i \cdot R_i \cdot \left(I - \frac{(t_0 - t_i) \cdot n_0^T}{d} \right) \cdot R_0^T \cdot K_0^T \qquad (1)$$

where $H_i(d)$ is the homography between the i^{th} feature map and the reference feature map at depth d. Moreover, K_i, R_i, t_i refer to the camera intrinsics and extrinsics with index 0 indicating the reference view, and n_0 is the principle axis of the reference camera. Since for the reference feature map itself the homography is the identity matrix, the original feature map is repeated on every plane.

Next, we can aggregate N feature volumes F_i to one cost volume C by using a variance based cost metric:

$$C = \frac{\sum_{i=1}^{N} (F_i - \bar{F}_i)^2}{N} \qquad (2)$$

where $\bar{F}_i$ is the average volume among all feature volumes.

We perform these operations on all of our feature extraction levels to acquire 4 cost volumes, each corresponding to a different scale.

C. COST VOLUME REGULARIZATION

Next, we decode (i.e. filter) our cost volumes in a coarse-to-fine manner and fuse each output with the finer scale raw feature volume. This is in contrast to [9], where the cost volumes themselves are not propagated.

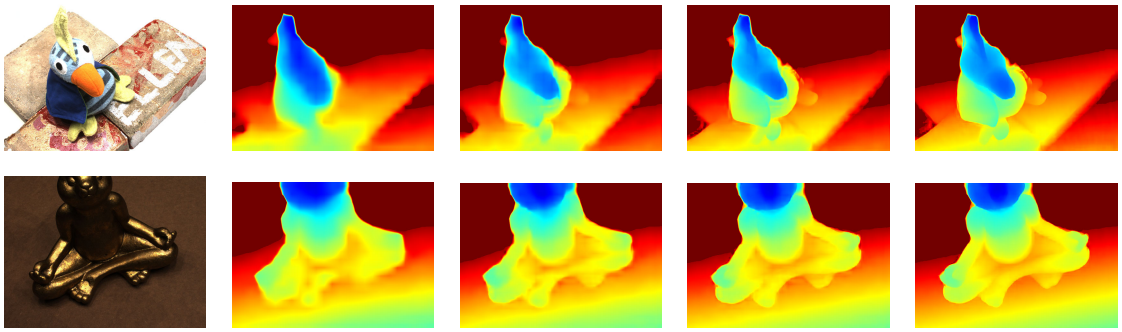
1) DECODER

We adapt the decoder structure used in [30] with 4 decoder blocks, each responsible for the output of our 4 stages. Each decoder block consists of 6 3D convolution blocks which in turn contain two 3D convolutions with a residual connection. As in the feature extraction (encoder) part, pyramid pooling is then used to make sure sufficient global context for the high-resolution input is captured. The decoder block then generates 2 outputs: 1) A cost volume, which will be fused with the input of the next stage. 2) A classified cost volume, which when put through a softmax layer and depth regression yields a depth map to initialize the feature volumes in the next stage. The classification is done via a 3D convolution layer, followed by a ReLU layer and another 3D convolution layer which maps every feature to 1 channel. By applying a softmax layer afterwards this can be seen as the probability for every depth hypothesis.

In the coarsest stage (i.e. stage 4), we generate the output only from the raw cost volume which covers the whole depth range of the input scene in only a few hypothesis planes. Subsequent decoder blocks use the bilinearly upsampled output fused with the corresponding raw cost volume.

2) CASCADING COST VOLUMES

Following [9], we build these ensuing cost volumes upon a narrower depth range based on the previous prediction using the cascading cost volume formulation. The main idea behind this concept is that if we have a network that estimates depth in a coarse-to-fine manner, we can take the coarse prediction as a prior for the next stage and only search hypothesis planes in its vicinity. This controls the cost volume size and drastically reduces memory requirements. One drawback of this method is that it is very dependent on the coarsest depth estimate. If the coarsest estimation is too far off the real depth value, there is no way for the network to predict the correct depth, resulting in a large error. This can especially be observed at object borders, where depth at the coarsest level is often ambiguous. However, if we later fuse multiple depth predictions into one point cloud the issue does not persist, since the wrong predictions are not consistent.



(a) Input (b) Stage 4 (c) Stage 3 (d) Stage 2 (e) Stage 1

FIGURE 5. Example depth map outputs of our network at all 4 stages. Note, how we already have a good rough estimate at the coarsest stage (stage 4). The depth map gets smoother and smaller details appear throughout the finer stages.

We estimate the depth on every scale by upsampling the classified cost volumes to the desired output size before using depth regression (see Fig. 5).

D. LOSS

To regularize the network, we apply a multi-scale loss:

$$L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3 + \lambda_4 L_4 \tag{3}$$

with

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \tag{4}$$

where L_k refers to the loss at stage k and λ_k is the corresponding loss weight. To be consistent with the encoder formulation, L_1 represents the loss on the finest scale, while L_4 is the loss on the coarsest scale. To account for the different scale levels we set:

$$\lambda_n = \frac{1}{4}\lambda_{n-1} \tag{5}$$

We calculate the loss at every stage as the mean absolute difference between predicted and ground truth depth map. This is compliant with the loss formulation used in [30].

IV. IMPLEMENTATION

We implement our HighRes-MVSNet in Pytorch. Our network is flexible to take in any input size as long as both, height and width, are divisible by 64. This is due to our coarsest scale being $\frac{1}{64}$ of the input size. Our dataloader will take care of this and also adjust the intrinsic camera parameters accordingly. The output size is also adaptable, since the network uses a cost volume scaling in its final stage. Through empirical evaluation we found that an output size that is equal to $\frac{1}{2}$ the input size is the most practical as it yields the best evaluation results while the point cloud fusion does not take an exaggerate amount of time.

A. DEPTH HYPOTHESIS PLANES

For every input image we have to define a depth range between the minimum depth d_{min} and the maximum depth d_{max}. We can usually extract this information with a Structure from Motion pipeline. In SfM, camera poses of images are recovered by matching sparse features. From the sparse feature information, we can determine the closest and furthest point of interest in the image. This effectively gives us the required depth range r.

We uniformly sample D depth hypotheses over this range. The finest depth difference between our hypothesis planes is then $\frac{r}{D}$. Since we are using the cascading cost volume formulation, we only need to cover a fraction of the depth range in every stage except stage 4, which covers the whole depth range. Each subsequent stage will have its hypothesis range defined by the previously predicted depth.

Let us define the number of hypothesis planes at stage k as h_k. In order to cover the whole scene h_k has to satisfy:

$$i_{k+1} \leq i_k \cdot h_k \tag{6}$$

where i_k is the depth interval between hypothesis planes at stage $k \leq 3$ defined as:

$$i_k = \frac{r}{D} \cdot 2^{k-1} \tag{7}$$

To get an estimate over the whole scene in the coarsest stage we set $i_4 = \frac{r}{h_4}$.

Furthermore, h_{k-1} can only differ from h_k by a factor of 2^n or $\frac{1}{2^n}$. This is due to the fact that the network has to up or downsample the cost volume of the previous stage to fuse it with the cost volume of the current stage.

B. TRAINING

For a fair comparison with other MVS methods, we firstly train HighRes-MVSNet on the DTU [1] training data generated by MVSNet [32] and set $D = 384$. We use Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and set the learning rate to 0.001. During training, we reduce the learning rate at epochs 10, 12 and 14 by 2 and train for a total of 16 epochs. We fix the number of input images to 3 and the input image resolution to 1600×1152. As the number of hypothesis planes for every stage we choose $h_4 = 16$, $h_3 = 16$, $h_2 = 16$, and $h_1 = 8$. The training takes approximately 5 days with a batch size of 2 on an NVIDIA GeForce GTX 1080Ti.

We also train our network from scratch on the recently released BlendedMVS [34] high-res dataset. This dataset offers 113 diverse scenes and images come at a resolution of 2048 × 1536. We use the same hyperparameters as for the DTU dataset, except for $D = 512$, $h_4 = 16$, $h_3 = 32$, $h_2 = 16$, and $h_1 = 8$. Furthermore we replace the batch normalization (BN) [13] with group normalization (GN) [28] as suggested in [34]. We further discuss this in Section V-C.

C. POINT CLOUD FUSION

Our network outputs a depth prediction in form of a depth map for every input image. However, depending on the image content, many of these predictions can be outliers since the pixel may belong to the background or an occluded area. We therefore check for geometric consistency before projecting every pixel into 3D space in order to obtain a dense point cloud. We do this by projecting a reference pixel p_{ref} through its depth d_{ref} to pixel p_i in a different view and then reproject p_i through d_i to obtain p_{reproj} and d_{reproj}. The depth is now 2 view consistent if it satisfies:

$$\|p_{ref} - p_{reproj}\| < \tau_1 \tag{8}$$

and

$$\|d_{ref} - d_{reproj}\| < \tau_2 \tag{9}$$

where τ_1 and τ_2 are threshold values for the pixel distance and the relative depth difference, respectively. We consider a pixel valid if its depth is consistent in at least n views. This number of consistent views is a parameter that can be adapted for different scenes, but in our experience should always be $n \geq 3$.

V. EVALUATION

We evaluate our HighRes-MVSNet on the well known DTU [1] and Tanks and Temples [18] benchmarks. Note, that the reported scores are very depended on the point cloud fusion algorithms and parameters used. We only use the simple geometric verification as described in Section IV-C.

A. RESULTS ON DTU DATASET

In the case of the DTU dataset every object lies approximately within the same distance and we set $d_{min} = 425mm$, $d_{max} = 1065mm$ and $D = 384$ for every scene. We use the network weights obtained from training on the DTU training set. For the point cloud fusion we set the number of consistent views to $n = 3$ and the thresholds to $\tau_1 = 0.25$ and $\tau_2 = 0.01$.

Quantitative results evaluated on the test set can be found in Table 1. We can see that our network achieves state-of-the-art results in terms of scene *completeness*, *accuracy* and the overall score. In terms of GPU memory consumption and runtime we outperform all other methods by a large margin when using the provided image size as input (see Tab. 2). For [4], [31]–[33] we show the values reported by [31] who executed the networks on an NVIDIA TITAN RTX. For [9], [35] and our method we show the values obtained ourselves by running the official evaluation code of baselines on an NVIDIA GeForce GTX 1080 Ti.

TABLE 1. Quantitative results on the DTU dataset. All scores are in *mm* and represent the mean average distance (lower is better). Ours(HR) scales the input images to a resolution of 3200 × 2368. Best results are shown in bold and the runner-ups are underlined.

	Method	Acc.	Comp.	Overall
Geometric	Furu [6]	0.613	0.941	0.777
	Tola [27]	0.342	1.190	0.766
	Camp [2]	0.835	0.554	0.695
	Gipuma [7]	**0.283**	0.873	0.578
	COLMAP [25], [26]	0.400	0.664	0.532
Learning	MVSNet [32]	0.396	0.527	0.462
	R-MVSNet [33]	0.383	0.452	0.417
	SurfaceNet [14]	0.450	1.040	0.745
	MVSCRF [29]	0.371	0.426	0.398
	Point-MVSNet [4]	0.342	0.411	0.376
	CasMVSNet [9]	0.346	0.351	<u>0.348</u>
	CVP-MVSNet [31]	<u>0.296</u>	0.406	0.351
	AttMVS [20]	0.383	**0.329**	0.356
	Fast-MVSNet [35]	0.336	0.403	0.370
	Ours	0.354	0.393	0.373
	Ours(HR)	0.346	<u>0.345</u>	**0.346**

To show that our network can easily handle high-resolution input, we also evaluate the dataset when scaling the input images to a resolution of 3200 × 2368 (2x original size) and setting $D = 768$. We retrain the network with GN and set $h_4 = 16, h_3 = 16, h_2 = 8, h_1 = 8$ due to memory constraints in the training phase. This will improve the results at the cost of increased GPU memory consumption (2585 MB) and runtime (0.34 seconds/image). We attribute this to the fact that we get finer depth estimates at the coarsest stage of our network. Effectively, the cost volume is then built at $\frac{1}{32}$ of the image size instead of $\frac{1}{64}$. Qualitative results can be found in Figure 6 and Figure 8.

B. RESULTS ON TANKS AND TEMPLES DATASET

To evaluate HighRes-Net on the Tanks and Temples benchmark, we use the weights obtained from training on the BlendedMVS dataset. Again, we upscale the input images by a factor of 2. Since all sequences in this benchmark consist of many images with large view overlaps, we adapt the point cloud fusion parameters and set the number of consistent views to $n = 5$ and the thresholds to $\tau_1 = 0.5$ and $\tau_2 = 0.01$.

Table 3 shows the quantitative results on the intermediate dataset. We achieve state-of-the-art results without the use of any advanced depth filtering method for point cloud fusion. The generated point clouds are shown in Figure 7.

C. ABLATIONS

To find the best network hyperparameters, we perform a set of ablation experiments. Results are shown in Table 4.

1) NUMBER OF HYPOTHESIS PLANES

We test different numbers of hypothesis planes in each stage and observe that this will only slightly influence the accuracy. As long as the requirements discussed in IV-A are met,

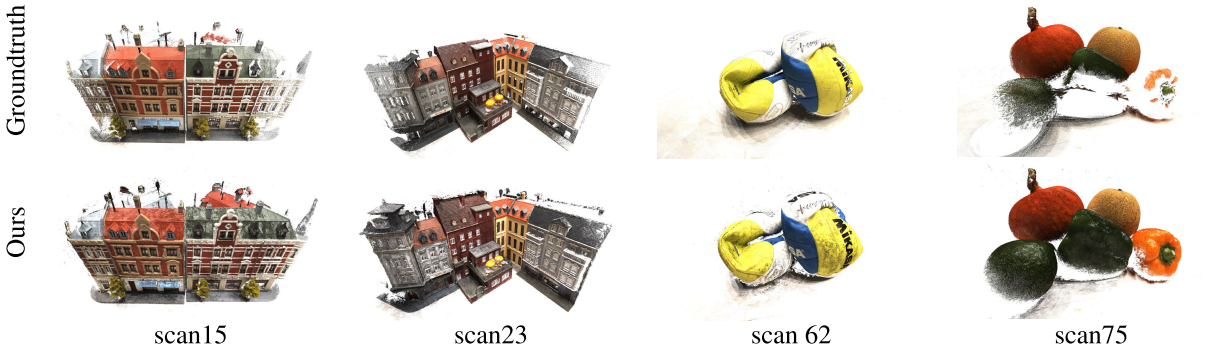


FIGURE 6. Qualitative results on the DTU dataset. We can see that our method is able to reconstruct even more details than provided by the ground truth models.

FIGURE 7. The qualitative results of the Tanks and Temples dataset.

TABLE 2. Results on the DTU dataset for an input image size of 1600 × 1152. Our method achieves comparable results with state-of-the-art methods in terms of accuracy and completeness while being at least 8× faster and using 6× less memory with respect to the second best performances. Best results are shown in bold and the runner-ups are underlined.

Method	Acc.(mm)	Comp.(mm)	Overall(mm)	GPU Mem.(MB)	Runtime(s)
MVSNet [32]	0.396	0.527	0.462	22511	2.76
R-MVSNet [33]	0.383	0.452	0.417	<u>6915</u>	5.09
Point-MVSNet [4]	0.342	0.411	0.376	13081	3.04
CasMVSNet [9]	0.346	**0.351**	**0.348**	9891	<u>0.85</u>
CVP-MVSNet [31]	**0.296**	0.406	<u>0.351</u>	8795	1.72
Fast-MVSNet [35]	<u>0.336</u>	0.403	0.370	7255	1.00
Ours	0.354	<u>0.393</u>	0.373	**1119**	**0.10**

the overall score on the DTU dataset only marginally changes. However, it does influence the memory consumption and runtime, especially in the training phase, since these numbers are directly related to the cost volume size.

2) BN vs GN

Although our network is very efficient regarding memory requirements in the evaluation phase, it still needs a considerable amount of GPU memory during training. Therefore, we are only able to train our network with a maximum batch size of 2. As we learn from [28], BN's error can increase rapidly with a small batch size due to inaccurate batch statistics estimation. On the other hand, GN stays stable over a wide range of batch sizes. Furthermore, GN enables us to train on a batch size of 1, thus further decreasing memory requirements for the training phase. We observe that BN works slightly better when we have a more constrained dataset (DTU). When training on the BlendedMVS dataset though, BN will lead to invalid values due to the aforementioned inaccurate statistics estimation.

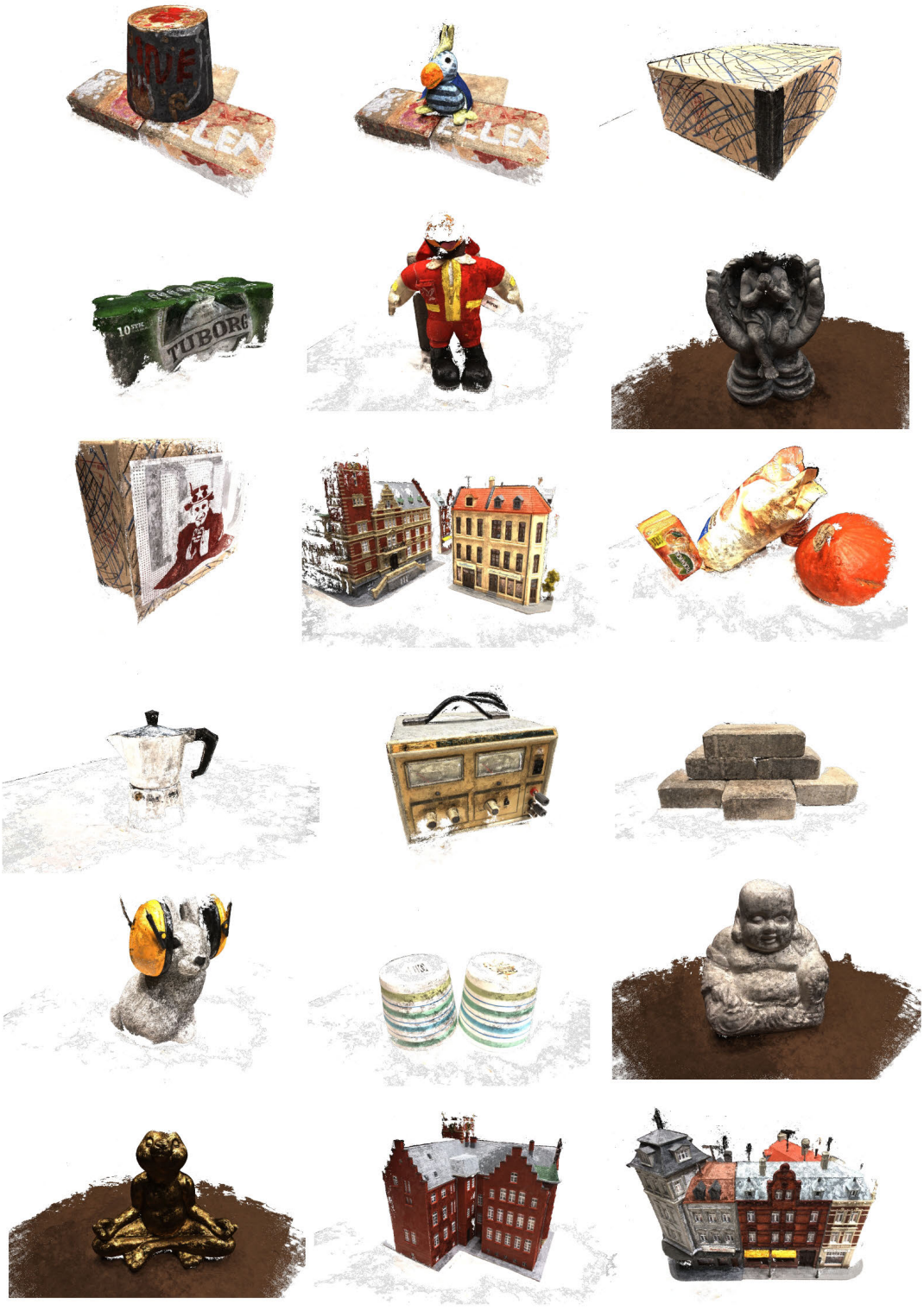


FIGURE 8. Further qualitative results on the DTU dataset.

3) OUTPUT SCALING

The finest cost volume produced by HighRes-MVSNet is at $\frac{1}{8}$ resolution of the input size. This is due to the fact that in the first stage of our feature extractor, all information is already encoded to this smaller resolution. For lower resolution images, like in the benchmark datasets, we then scale the cost volume up to the desired output size before regressing the depth. We find that an output scale of 0.5 uses less memory

TABLE 3. Results on the Tanks and Temples intermediate dataset of state-of-the-art MVS and our method. Precision and recall is combined as *f-score* (higher is better).

Method	Mean	Family	Francis	Horse	Lighthouse	M60	Panther	Playground	Train
COLMAP [25], [26]	42.41	50.41	22.25	25.63	56.43	44.83	46.97	48.53	42.04
MVSNet [32]	43.48	55.99	28.55	25.07	50.79	53.96	50.86	47.90	34.69
R-MVSNet [33]	48.40	69.96	46.65	32.59	42.95	51.88	48.80	52.00	42.38
Point-MVSNet [4]	48.27	61.79	41.15	34.20	50.79	51.97	50.85	52.38	43.06
AttMVS [20]	60.05	73.90	62.58	44.08	64.88	56.08	59.39	63.42	56.06
CasMVSNet [9]	56.42	76.36	58.45	46.20	55.53	56.11	54.02	58.17	46.56
CVP-MVSNet [31]	54.03	76.50	47.74	36.34	55.12	57.28	54.28	57.43	47.54
MVSCRF [29]	45.73	59.83	30.60	29.93	51.15	50.61	51.45	52.60	39.68
Fast-MVSNet [35]	47.39	65.18	39.59	34.98	47.81	49.16	46.20	53.27	42.91
Ours	49.81	66.62	44.17	30.84	55.13	53.20	50.32	55.45	42.73

TABLE 4. Ablations of HighRes-MVSNet on the DTU dataset for an input image size of 1600 × 1152. Depth hypotheses (i.e. number of planes) are ordered from coarse to fine (i.e. h_4, h_3, h_2, h_1). Scale refers to the corresponding output depthmap size and Prop. indicates that the cost volume is propagated through the different stages.

Depth Hypos	Norm.	Scale	Prop.	GPU Mem.(MB)	Runtime(s)	Acc.	Comp.	Overall(mm)
32,16,8,4	BN	0.5	✓	1171	**0.09**	0.351	0.406	0.379
32,16,16,4	BN	0.5	✓	1173	0.10	0.353	0.399	0.376
16,16,16,8	BN	1.0	✓	1699	0.12	**0.320**	0.438	0.379
16,16,16,8	BN	0.5	✓	1119	0.10	0.354	**0.393**	**0.373**
16,16,16,8	GN	0.5	✓	**1089**	0.15	0.365	0.396	0.380
16,16,16,16	GN	0.5		1099	0.13	0.358	0.459	0.409
16,32,16,8	GN	0.5	✓	1205	0.16	0.355	0.432	0.393
32,32,32,32	GN	0.5	✓	2059	0.21	0.365	0.401	0.387

while also producing a better overall score. Note, that for high-resolution images we do not always require full scale depth maps to generate very accurate and complete 3d models since we can still recover enough points from the lower scale depth map.

VI. CONCLUSION

We have presented HighRes-MVSNet, a deep CNN especially aimed at high-resolution data. We control the cost volume size by encoding features to a lower resolution and narrowing down the depth range search through coarse predictions in the decoder stages. This allows us to massively reduce GPU memory requirements and runtime while still achieving state-of-the-art or even better results.

Since this network is aimed at high-resolution input, we have to upscale the available benchmark input images in order to receive the best results. In the future, we will look into adapting the network structure to use finer scale cost volumes, thus removing the need for this unnecessary input upscaling step, to increase the accuracy on datasets like Tanks and Temples.

REFERENCES

[1] H. Aanás, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 153–168, Nov. 2016.

[2] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla, "Using multiple hypotheses to improve depth-maps for multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2008, pp. 766–779.

[3] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5410–5418.

[4] R. Chen, S. Han, J. Xu, and H. Su, "Point-based multi-view stereo network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1538–1547.

[5] X. Du, T.-Y. Lin, P. Jin, G. Ghiasi, M. Tan, Y. Cui, Q. V. Le, and X. Song, "SpineNet: Learning scale-permuted backbone for recognition and localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11592–11601.

[6] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1362–1376, Aug. 2010.

[7] S. Galliani, K. Lasinger, and K. Schindler, "Gipuma: Massively parallel multi-view stereo reconstruction," *Publikationen Deutschen Gesellschaft Photogrammetrie, Fernerkundung Geoinfor.*, vol. 25, nos. 361–369, pp. 1–2, 2016.

[8] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[9] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan, "Cascade cost volume for high-resolution multi-view stereo and stereo matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2495–2504.

[10] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "MatchNet: Unifying feature and metric learning for patch-based matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3279–3286.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2017, pp. 2961–2969.

[12] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "DeepMVS: Learning multi-view stereopsis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2821–2830.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[14] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang, "SurfaceNet: An end-to-end 3D neural network for multiview stereopsis," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2307–2315.

[15] A. Kar, C. Häne, and J. Malik, "Learning a multi-view stereo machine," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 365–376.

[16] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-End learning of geometry and context for deep stereo regression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 66–75.

[17] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi, "Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 573–590.

[18] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Jul. 2017.
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[20] K. Luo, T. Guan, L. Ju, Y. Wang, Z. Chen, and Y. Luo, "Attention-aware multi-view stereo," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1590–1599.
[21] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5695–5703.
[22] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3061–3070.
[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[24] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proc. German Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2014, pp. 31–42.
[25] J. L. Schonberger and J.-M. Frahm, "Structure-from-Motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4104–4113.
[26] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 501–508.
[27] E. Tola, C. Strecha, and P. Fua, "Efficient large-scale multi-view stereo for ultra high-resolution image sets," *Mach. Vis. Appl.*, vol. 23, no. 5, pp. 903–920, Sep. 2012.
[28] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.
[29] Y. Xue, J. Chen, W. Wan, Y. Huang, C. Yu, T. Li, and J. Bao, "MVSCRF: Learning multi-view stereo with conditional random fields," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4312–4321.
[30] G. Yang, J. Manela, M. Happold, and D. Ramanan, "Hierarchical deep stereo matching on high-resolution images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5515–5524.
[31] J. Yang, W. Mao, J. M. Alvarez, and M. Liu, "Cost volume pyramid based depth inference for multi-view stereo," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Oct. 2020, pp. 4877–4886.
[32] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 767–783.
[33] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan, "Recurrent MVSNet for high-resolution multi-view stereo depth inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5525–5534.
[34] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "BlendedMVS: A large-scale dataset for generalized multi-view stereo networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1790–1799.
[35] Z. Yu and S. Gao, "Fast-MVSNet: Sparse-to-Dense multi-view stereo with learned propagation and gauss-Newton refinement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1949–1958.
[36] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, Jan. 2016.

RAFAEL WEILHARTER received the B.Sc. and M.Sc. degrees in telematics from the Graz University of Technology, in 2014 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Institute of Computer Vision and Graphics (ICG). His master's studies were focused on computer vision and autonomous robots. His research interests include 3D reconstruction and deep learning.

FRIEDRICH FRAUNDORFER (Member, IEEE) received the Ph.D. degree in computer science from the Graz University of Technology (TU Graz), Austria, in 2006. He had a postdoctoral stay at the University of Kentucky, the University of North Carolina at Chapel Hill, and ETH Zürich. From 2012 to 2014, he was the Deputy Director of the Chair of Remote Sensing Technology, Technical University of Munich. He is currently an Associate Professor with the Institute of Computer Graphics and Vision, TU Graz. His main research interests include 3D computer vision, robot vision, and machine learning techniques.