HAMMER: Learning Entropy Maps to Create Accurate 3D Models in Multi-View Stereo

Rafael Weilharter
rafael.weilharter@icg.tugraz.at

Friedrich Fraundorfer
fraundorfer@icg.tugraz.at

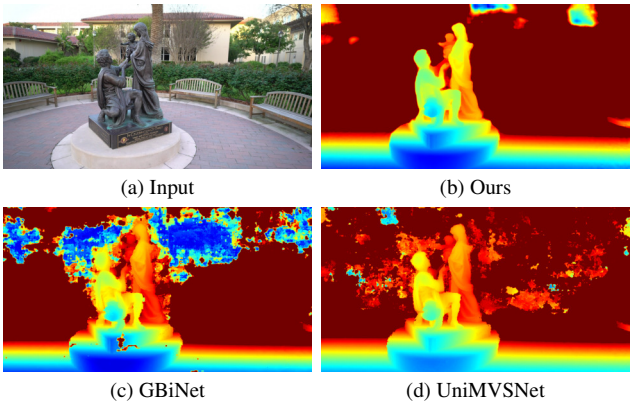
Institute of Computer Graphics and Vision
Graz University of Technology

Abstract

While the majority of recent Multi-View Stereo Networks estimates a depth map per reference image, their performance is then only evaluated on the fused 3D model obtained from all images. This approach makes a lot of sense since ultimately the point cloud is the result we are mostly interested in. On the flip side, it often leads to a burdensome manual search for the right fusion parameters in order to score well on the public benchmarks. In this work, we tackle the aforementioned problem with HAMMER, a Hierarchical And Memory-efficient MVSNet with Entropy-filtered Reconstructions. We propose to learn a filtering mask based on entropy, which, in combination with a simple two-view geometric verification, is sufficient to generate high quality 3D models of any input scene. Distinct from existing works, a tedious manual parameter search for the fusion step is not required. Furthermore, we take several precautions to keep the memory requirements for our method very low in the training as well as in the inference phase. Our method only requires 6 GB of GPU memory during training, while 3.6 GB are enough to process 1920×1024 *images during inference. Experiments show that HAMMER ranks amongst the top published methods on the DTU and Tanks and Temples benchmarks in the official metrics, especially when keeping the fusion parameters fixed.*

1. Introduction

The goal of Multi-View Stereo (MVS) is to obtain a 3D reconstruction of an observed scene from multiple overlapping images. As a prerequisite MVS requires known camera parameters which can be obtained via Structure from Motion (SfM). The idea is then to match pixels across views to obtain a depth estimate. Traditional methods [5, 6, 16], that use hand-crafted similarity metrics for this task have been replaced by learning-based methods over the recent years [12, 13, 18, 23, 24, 27]. Amidst the plethora of new approaches, a considerable number adheres to a similar concept: At first, dense features are generated through a feature extraction network. Subsequently, these features are aggregated into a cost volume following the plane sweep algorithm [4]. Lastly, the cost volume undergoes regularization to derive the final output in the form of a depth map.



(a) Input (b) Ours (c) GBiNet (d) UniMVSNet

Figure 1. The illustration shows a comparison of network outputs for different methods. We can see that our network is able to produce very smooth depth maps with low noise.

Despite the fact that the output is first a depth map, the most common benchmarks today [1, 9] only evaluate point clouds, i.e. 3D models instead of depth maps. While the depth map creation is handled by a neural network, point cloud are still produced in a classical manner by checking for geometric and photo-metric consistency. A photometric mask is often obtained by evaluating the probability weights of the network output, which is not explicitly learned [7, 19, 23]. For the geometric filtering, different fusion frameworks exist that project pixels into 3D space and have several similar parameters which can be tuned: 1) Interval scale. A parameter to influence the depth range in every view. 2) Consistent number of views. The number

of source images that have to confirm the depth estimate to guarantee geometric consistency. 3) Back-projection error in pixels. The distance error when projecting one pixel into another view and back. 4) Relative depth error. The relative error of the back-projection in depth direction.

Almost every MVS-Method needs to manually tweak these parameters - often for every individual scene - which can be a tedious task. Additionally, fusion is a separate, non-learning step that makes it hard to evaluate the true performance of a network. In contrast to the previously mentioned methods, we are able to apply a single set of fusion parameters that only relies on a geometric two-view consistency check.

In this work, we are introducing our Hierarchical And Memory-efficient MVSNet with Entropy-filtered Reconstructions (HAMMER) solution to efficiently obtain 3D models from 2D images without the need for a tedious fusion parameter search. Our framework has very low GPU requirements for training (about 6 GB) and is able to produce very clean depth maps (see Fig. 1). Our main contributions can be summarized as follows:

- A training method to produce a filter mask in addition to the required depth map. In contrast to many other methods, we directly learn these filter masks by applying our novel entropy-based loss at the output of a dedicated convolutional network block. This alleviates the problem of searching for the best parameters in the fusion step.

- We introduce randomized matched patches for the training phase. This allows us to use the full resolution of any available dataset without removing information by resizing images.

- We extend the multi-stage network design introduced by GBi-Net [12] by an adjustable interval parameter ψ which enables the network to be trained on any arbitrary depth resolution without affecting the network's memory requirements.

- Additional to the traditional benchmark evaluations, we provide comparisons between network performances when the fusion parameters are kept at a fixed setting. We also conduct extensive evaluations to show that HAMMER ranks amongst the top methods on the DTU [1] and the more challenging Tanks and Temples [9] benchmarks when compared to the officially reported scores.

2. Related Work

Multi-View Stereo has been an integral part of the 3D computer vision field for decades. Traditional approaches like COLMAP [15, 16] or Gipuma [6] match handcrafted features between images to estimate a dense 3D structure of the observed scene. While these methods performed well in terms of accuracy, they were clearly lacking in completeness, often struggling with reconstructing low-textured regions, non-Lambertian surfaces or matching noise.

In recent times, learning-based MVS methods have greatly outperformed the traditional approaches and still keep improving every year. The conventional approach in these deep learning-based techniques employs Deep CNNs to predict 2D depth maps. A novel development involves the widespread application of 3D cost volumes from image features [18, 23, 26]. Pioneered by MVSNet [23], these methods construct a 3D cost volume through feature warping and then employ 3D CNNs to regularize it for depth regression or classification. However, an issue inherent in the vanilla MVSNet is the substantial memory consumption tied to the 3D cost volume which, in turn, depends on image size and depth resolution. In response, various methods have been proposed in recent years to overcome this problem. Recurrent network architectures [21, 24] sequentially regularize the 2D cost maps along the depth direction to reduce the memory consumption. Cascading architectures [7] adhere to a gradual refinement strategy, moving from coarse-to-fine over multiple stages of different scales. As an alternative approach, PatchmatchNet [18] improves the Patchmatch [2] core algorithm with a novel and learned adaptive propagation and evaluation scheme for each iteration. It deliberately reduces the heavy regularization present in 3D cost volumes to create an efficient model, although this entails a trade-off between efficiency and accuracy. Recently, GBiNet [12] formulates MVS as a binary search problem, reducing the depth hypothesis requirement per stage to 4. To ensure a fine enough depth resolution, each feature scale is regularized by the same 3D CNN two times with a subsequently smaller resolution. The authors also propose to randomly crop the input images for training on the DTU dataset. This reduces memory consumption without the need to downscale images, thus keeping the full information content of the dataset. In our work we build on several of these insights and design a deep neural network that is able to handle images on a very adaptable memory requirement, enabling training and evaluation on most consumer grade GPUs.

While a lot of progress has been made for depth map estimation networks, basically only 3 approaches have been used for creating a 3D model from those depth maps: 1) The fusion method 'fusible' of Gipuma [6] written in `C++`. 2) A similar version of 1 but for `Python` proposed by MVSNet [23]. 3) The dynamic consistency checking variant of 2 proposed by D^{2}HC-RMVSNET [21].

As mentioned in 1, all of these have similar parameters that can be adjusted. To the best of our knowledge, all of the mentioned works have to manually adapt the fusion pa-

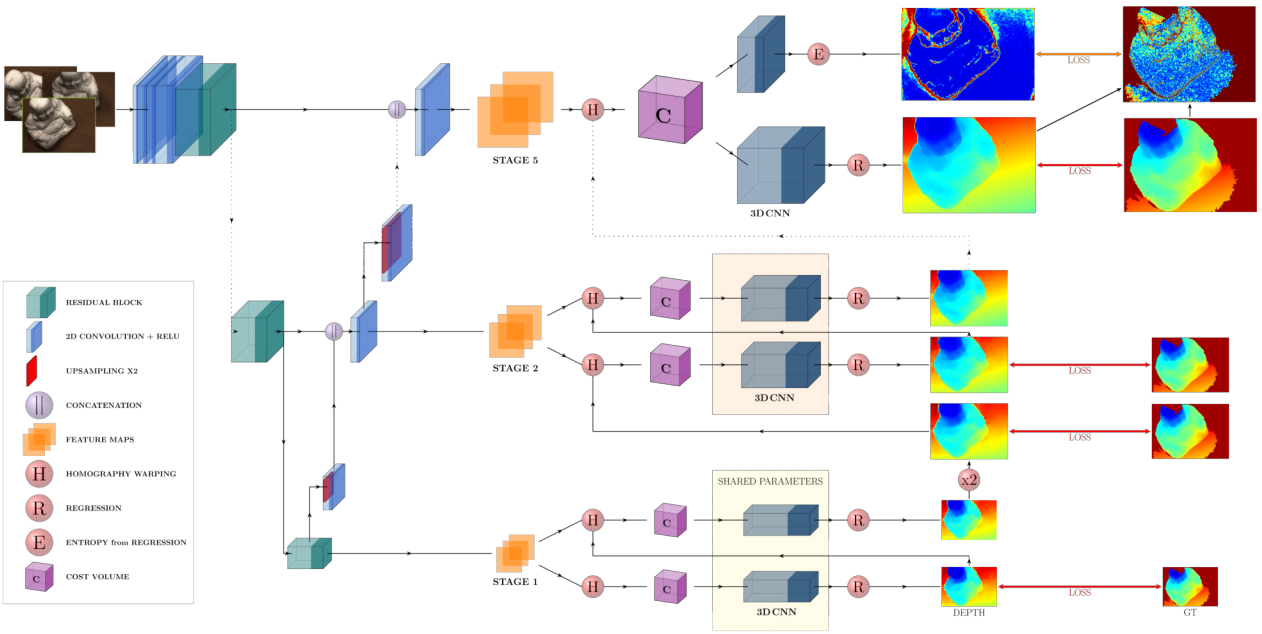


Figure 2. Overview of the proposed HAMMER architecture: We first extract features at multiple scales by applying a UNet to a given set of images. The obtained feature maps are then aggregated into a cost volume through a variance based cost metric and homography warping. We use a 3D CNN consisting of 10 layers to regularize the cost volume at each scale. Finally, we can estimate the depth from the cost volume via regression. Each depth estimate at a lower stage is used to initialize the depth hypothesis of the next stage. In order to save memory, we pass the aggregated feature maps through the same 3D CNN twice, but apply a different homography warping each time, which yields a finer depth resolution.

rameters or even switch the fusion framework completely when applying their method to a different dataset or scene. With HAMMER we want to propose a method that works well across different scenes and datasets without the need to adjust any screws.

3. Method

This section introduces the detailed architecture of HAMMER that is depicted in Figure 2. Before actually training the network, we introduce a pre-processing technique that allows for a more memory-flexible training of MVS networks which we refer to as *randomized matched patches*. For the network itself, we adapt several components from previous networks that showed great performance: Firstly, we perform feature extraction on the reference image and its n source images with a hierarchical UNet [14], also known as Multi-scale Feature Net, similar to previous networks like GBi-Net [12] or ATLAS-MVSNet [20]. We design our network in such a way that the output resolution of the highest layer in the hierarchy is $\frac{1}{2}$ of the input image size. We then upscale the final cost volume after regularization to achieve the same resolution for the depth map as the input image. Secondly, homography-warping and depth initialization is done with the cascading cost volume formulation [7], as is practice in most modern MVS methods. However, we extend this algorithm and process each feature stage twice, similar to GBi-Net. Thirdly, we regularize the 2 cost volumes of each feature stage with 2 different depth resolutions defined by the adjustable interval parameter ψ which allows us to set an arbitrary depth resolution that is independent of memory requirements.

3.1. Pre-processing: Randomized Matched Patches

An issue with current available data is that, while relatively high resolution images (e.g. DTU [1] has 1600×1200) are available, it is often too memory intensive to use the full resolution in the training phase. GBiNet [12] proposes to crop random patches for training to learn better features without increasing the training overhead and simultaneously keeping the memory consumption in check. While this works good enough for a well defined dataset like DTU, it fails to get overlapping patches in datasets with heavily varying viewpoints like BlendedMVS [25]. Hence, we introduced randomized matched patches: From any given reference image, we randomly crop a part of the image during training to an arbitrary size that fits the network (in our case 512×512). As the ground-truth depth map and camera parameters are known, we can project the pixels of the reference patch into the source images. To find a corresponding

Figure 3. Randomized matched patches. If the viewpoint changes significantly between reference (left) and source (right) image, randomly cropping the same patch in both images might have little to no overlap (red square). Our RMP method selects a random patch only in the reference image and projects several points (red crosses) into the source image to find a patch with guaranteed overlap (green square).

patch, we first project p pixels around the center pixel and check if at least 25% are valid, i.e. are not outside the source image or contain an invalid depth value. We then calculate the centroid of the valid point to obtain the center pixel of the source patch (see Fig. 3). This process can be done online and is repeated for all source images and guarantees a significant overlap between the patches (see Fig. 3). If no valid matches can be found, a new random patch is chosen in the reference image. In the rare case that this process fails 20 times, we skip the image.

3.2. Feature Extraction

We run each image patch through a U-Net [14] to hierarchically extract features at 5 stages of different resolutions. We first apply 4 convolutional layers that sequentially increase the number of channels from 3 (RGB image) to 32. For down-scaling the features we use a residual block consisting of 2 convolutional layers with stride set to 2 in the first layer. It follows that our finest grained feature map has $\frac{1}{2}$ the size of the full resolution input image. As can be seen in Figure 2, the features from the lowest scale are up-scaled and concatenated to obtain the feature map of each stage. This design ensures that our largest cost volume will only need to cover half of the input image resolution before regularization, which helps keeping the memory requirements for the 3D CNN low.

3.3. Cost Volume Assembly

Following previous approaches [3, 7, 22, 24], we warp features into a single cost volume utilizing the differential homography and aggregate them via variance-based cost metric to allow for an arbitrary number of input images. The differential homography is defined as:

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

where $H_i(d)$ is the homography between the i^{th} source feature map and the reference feature map (index 0) at depth d. The camera intrinsics and extrinsics are defined by the parameters $K_i, R_i, \mathbf{t_i}$ while $\mathbf{n_0}$ is the principle axis of the reference camera and I is the identity matrix.

To keep the cost volume size in check, we adapt the strategy of GBiNet [12] that uses two cascading cost volumes with only 4 depth hypothesis for each feature stage. However, their approach halves the depth interval after every pass through the regularization network which results in a stiff network architecture with a fixed depth resolution. We introduce an interval parameter ψ that allows us to freely select the depth resolution without changing the memory requirements of the network. After the first depth interval is found by dividing the full depth range by 4 (from the 4 depth hypothesis that we have for each cost volume), we can simply multiply this parameter with the previous depth interval to get the new one. To achieve our goal of refining the depth interval in every stage, it follows that $0 < \psi < 1$. We found that a value of $\psi = 0.55$ works well for our network design with 5 stages (each stage applies the parameter twice), which results in a depth resolution of $0.25 \cdot 0.55^{(10-1)} \approx 0.001$ times the full depth range.

3.4. Network Outputs

In order to remove noise from the cost volume, we pass it through a regularization network that consists of 5 residual blocks, each containing 2 convolutional layers. The network outputs 2 depth maps which are obtained via regression and the *soft argmin* operation [8] for every feature stage:

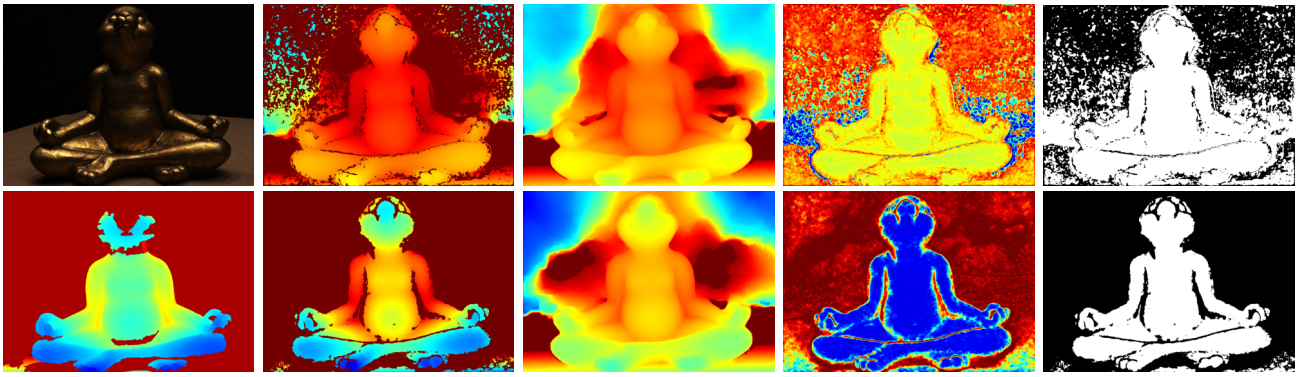
$$\textit{soft argmin} := \sum_{d=1}^{d_{max}} d \times \sigma(-c_d), \quad (2)$$

where d_{max} is the maximum depth value of the current depth range, c_d is the predicted cost and $\sigma(\cdot)$ is the *softmax* operation.

These depth maps are used to initialize the depth hypothesis for each subsequent stage and to calculate the loss. In the final stage, we also calculate an entropy map through 2 separate residual blocks. The idea is that a high value of entropy signals that the network applies equal probability to all depth hypothesis, indicating a high uncertainty. In the training stage, we enforce this behavior by applying an additional loss.

3.5. Loss Function

At first, we calculate the loss for each depth map. Because of the two different depth ranges for each feature map scale, we obtain 10 depth maps from 5 stages. Since the initialization of the finer cost volume has to be done with an up-scaled version of the depth map we also apply the loss to the up-scaled version. The loss is then accumulated over



(a) Input + GT (b) Output (c) Depth (d) Entropy (e) Mask

Figure 4. Comparison between training the network with and without the entropy loss. The input and ground-truth for both networks is depicted in (a). **Top:** Output from the proposed architecture (b), but without applying the entropy loss in the training. To calculate the mask (e), we use the naturally obtained entropy (d) from the regression of the depth map (c). **Bottom:** Our HAMMER method is able to produce very clean masks for valid values in the obtained depth map. The entropy map is taken from the separate output branch.

our 5 feature stages as:

$$L_d = \sum_{k=1}^{5} l_k, \tag{3}$$

with

$$l_k = \lambda_k \cdot (\|D_k - D_{k,gt}\|_1 + 2\|D_{k,up_2} - D_{k+1,gt}\|_1), \tag{4}$$

where λ_k is a weight factor that we increase by a factor of 2 every stage, D_k is the output depth map of the current stage, D_{k,up_2} is its up-scaled version and $D_{k,gt}$ is the ground-truth depth map.

As previously mentioned, for the final output we also calculate an entropy map. However, as our network is trained via regression rather than classification, there is no guarantee that entropy is low for good depth estimates. This is due to the fact that a good depth estimate could also be achieved by an equal distribution of probabilities, which is just treated as a different weighting in regression. However, we want to encourage the network to apply a single strong probability to the correct depth estimate bringing this approach closer to a classification task.

In order to enforce a low entropy on a correct depth estimate, we apply an additional loss between entropy and the depth error map $M_E = |D_k - D_{k,gt}|$ that we can obtain by calculating:

$$L_e(x,y) = \begin{cases} \epsilon_{max} - \epsilon(x,y) & \text{if } M_E(x,y) > \delta_t \\ \epsilon(x,y) & \text{else} \end{cases} \tag{5}$$

where $L_e(x,y)$ is the loss at pixel position (x,y), $\epsilon(x,y)$ is the entropy value and δ_t is a threshold value for the error in the depth map. The entropy at each pixel position $\epsilon(x,y)$ is defined as:

$$\epsilon(x,y) = \sum_{j=0}^{h-1} -p_j(x,y) \cdot log(p_j(x,y)), \tag{6}$$

where h is the number of plane hypothesis and $p_j(x,y)$ is the probability the network attributes to each hypothesis plane j. We can obtain the theoretical ϵ_{max} by setting the same probability to all hypothesis. The loss for the entropy over the whole image is the mean value over all positions.

Finally, the total loss L is calculated as:

$$L = L_d + L_e \cdot \delta_t, \tag{7}$$

where the threshold δ_t acts as a weighting factor to balance the losses. We calculate the threshold value δ_t in such a way that it is close to the finest depth resolution in each view.

3.6. Post-Processing

As most networks, the final output of our network is a depth map. However, benchmarks usually only evaluate point clouds, i.e. fused 3D models. We can obtain a point cloud by projecting the pixels of the depth maps into 3D space and fusing them together. This is a very crucial step in MVS and benchmark results depend heavily on the correct choice of fusion parameters. Most methods find these parameters empirically by trying different settings, which can often take a lot of time and is infeasible if many different scenes have to be evaluated. We propose a fair comparison by only using our entropy masks and a simple two view consistency check with fixed parameters which directly evaluates the quality of depth maps. In contrast to the majority of State-of-the-Art MVS networks, we are able obtain high quality point clouds by using the same settings for every dataset and scene (see Fig. 5).

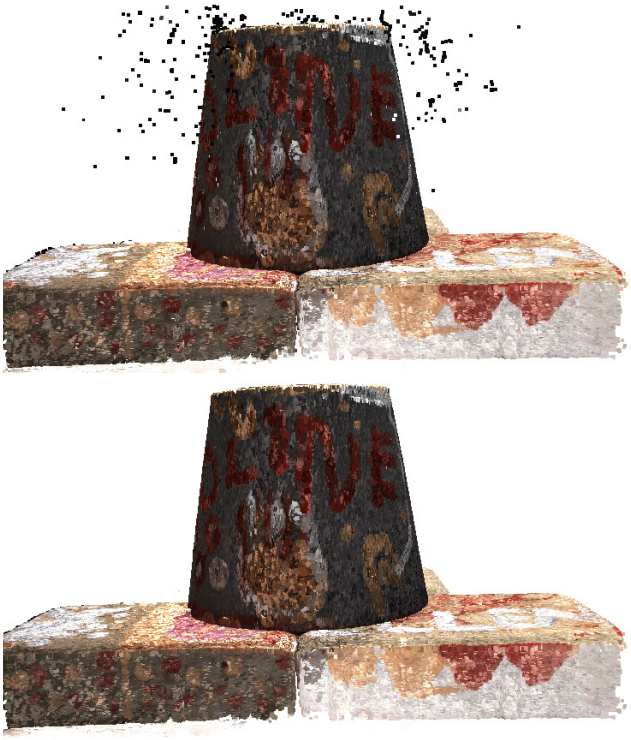


Figure 5. Point cloud comparison. **Top:** The obtained point cloud when applying the non-learned entropy mask. **Bottom:** By applying our learned entropy mask we can almost completely remove noisy points.

4. Implementation

We set the number of views during training to $N = 5$ and train for 18 epochs on the DTU training data. By setting the input patches to 512x512, the training of our method only requires about 6 GB of memory. As is common practice for a better generalization, we fine-tune our model from the 10^{th} epoch DTU [1] training for another 12 epochs on the BlendedMVS dataset [25]. The network is optimized via Adam optimizer in Pytorch with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and an initial learning rate of 0.0005, which is down-scaled by a factor of 2 after epochs 10, 12 and 14. We are able to train the network on a single Nvidia GeForce GTX 1080 Ti.

We tried several values for the depth interval parameter ψ, but found that it will not influence the final results too much as long as it is not too high (resulting in a very coarse depth resolution) or too low (will not cover enough depth when going from one stage to the other). We keep $\psi = 0.55$ for all our results presented here.

For testing we need 3.1 GB for the DTU images (1600×1152) and 3.6 GB for the Tanks and Temples images (1920×1024). We use $N = 5$ for the evaluation on DTU, but noticed an increase in performance when using $N = 10$ for Tanks and Temples. This can mainly be attributed to the increased number of images per scene and the smaller changes in viewpoints between those images.

Figure 6. Example point cloud reconstructions of our method. **Top:** DTU dataset. **Bottom:** Tanks and Temples dataset.

For the fusion we use the implementation of MVS-Net [23] and keep the parameters fixed for all scenes. We use an interval scale of 1.0 to cover the full depth range of the scene. Consistent number of views is set to 2 which means we are only checking if a single source image pixel confirms the depth estimate in the reference image. Note that this is quite an uncommon choice especially for the Tanks and Temples dataset where other methods often use 6+ views for geometric verification to avoid outlier points in the 3D model. We believe that this is a strong point for our filter masks obtained from the learned entropy. By being able to keep this parameter low we can reconstruct 3D points in many different scenes even when there is very little overlap between images. Back-projection error and relative depth error are fixed at 0.2 pixels and 0.001 respectively.

5. Results

All results are obtained by using the same fusion parameters. The only parameter that is adjusted is the number of input images for the network. As discussed in the previous section, the results tend to get slightly better when using more neighboring views for datasets that only show small changes in different views. Qualitative results of our method are shown in Figure 6.

5.1. Evaluation on DTU

For this evaluation we use the model trained purely on the DTU training set for 18 epochs. We compare our results on the DTU benchmark quantitatively to traditional and recent learning-based methods in Table 1. The average distance for every estimated 3D point from a ground truth point is reflected in the *Acc.* (accuracy) score. If we

Method	Acc.	Comp.	Overall
Gipuma [6]	0.283	0.873	0.578
COLMAP [15, 16]	0.400	0.664	0.532
MVSNet [23]	0.396	0.527	0.462
R-MVSNet [24]	0.383	0.452	0.417
CasMVSNet [7]	0.346	0.351	0.348
PatchmatchNet [18]	0.427	0.277	0.352
EPP-MVSNet [10]	0.413	0.296	0.355
ATLAS-MVSNet [20]	**0.278**	0.377	0.327
DELS-MVS [17]	0.342	0.284	0.313
UniMVSNet [13]	0.352	0.278	0.315
GBiNet* [12]	0.312	0.293	0.303
GeoMVSNet [27]	0.331	**0.259**	**0.295**
CER-MVS [11]	0.359	0.305	0.332
HAMMER (Ours)	0.326	0.270	0.298

Table 1. Quantitative results on the DTU test dataset. All scores are in mm and represent the mean average distance (lower is better). Best results are shown in bold and the runner-ups are underlined. GBiNet has been adjusted to use the same fusion parameters for all scenes for a fair comparison with all other methods.

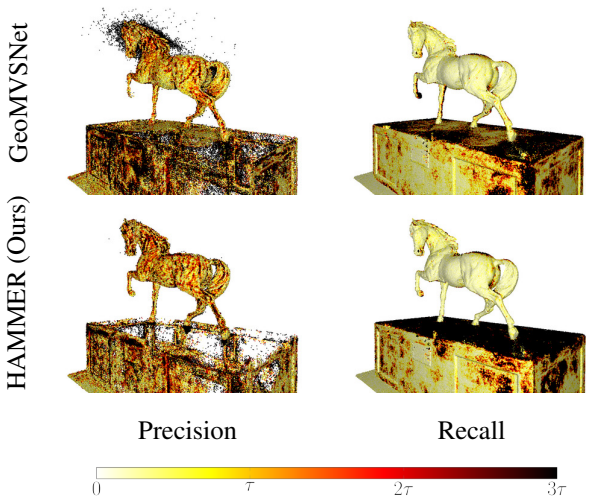


Figure 7. Qualitative comparison between GeoMVSNet [27] and our method on the Horse scene of Tanks and Temples. The color indicates the distance to the ground truth with $\tau = 9mm$. Our method shows significantly less outliers in terms of precision.

go into the other direction, i.e. the average distance for every ground truth 3D point from a generated point, we can obtain the *Comp.* (completeness) score. The *overall* score displays the mean between the two. Our approach demonstrates great performance, only being surpassed by the recently released GeoMVSNet [27].

5.2. Evaluation on Tanks and Temples

While it is widely accepted and common practice that the same fusion parameter settings are used for the whole dataset on DTU, this is not applied for the Tanks and Temples data.

At first, we compare the results published in the respective papers in Table 2. While our method does not reach the F-scores of GeoMVSNet [27], it produces significantly less outliers when compared qualitatively in Figure 7. We found that most methods use greatly varying parameters: While this is not an exhaustive list, we investigated the 4 main parameters mentioned in Section 1: 1) Interval Scale. 2) Consistent number of views. 3) Back-projection error in pixels. 4) Relative depth error.

Most of the time, the interval scale is set to 1.0 but many methods also use 1.06 to increase the performance on the benchmark. We have found that for the consistent number of views in the DTU dataset, usually a low number (often 3) is chosen, while for Tanks and Temples 6 or more might be applied. For the pixel distance most methods use 0.25 - 0.5 pixels for DTU, but to 2 - 3 pixels for Tanks and Temples. The relative depth error is often set several magnitudes lower in Tanks and Temples. Some methods, e.g. UniMVSNet [13], even use different filtering methods for different datasets, which may take several hours to complete.

We argue that for a fair network comparison, fusion parameters should stay at a constant setting. Therefore we compare HAMMER to high performing methods on the benchmark when settings are kept consistent throughout the dataset in Table 3. We can see that our approach performs exceptionally well without the need to change the fusion settings from the DTU evaluation. Note that we apply the average setting as suggested by the authors for the Tanks and Temples dataset alone without taking the settings from DTU into account, which usually differ greatly. However, we had to limit the selection due to code availability and hardware requirements. For example the code for GeoMVSNet [27] has not been made public yet and CER-MVS [11] needs a GPU with at least 24 GB of memory.

Regarding the Tanks and Temples advanced dataset, HAMMER performs reasonably well without adjusting the fusion parameters despite the fact that this dataset is very challenging. For most methods fusion parameters have to be set with a very large variation between scenes: Consistency checks can range from a single image up to 8 or more. Thresholds for the depth error are magnitudes apart. Confidence thresholds get dropped to a very low value to ensure higher completeness.

5.3. Ablations

In Figure 4 we can see the effects of the entropy training and filtering. We establish a baseline by training HAMMER without the entropy loss and compare it to several selected

	Intermediate									Advanced						
Method	**Mean**	**Fam**	**Fra**	**Hor**	**Lig**	**M60**	**Pan**	**Pla**	**Tra**	**Mean**	**Aud**	**Bal**	**Cou**	**Mus**	**Pal**	**Tem**
COLMAP [15, 16]	42.41	50.41	22.25	25.63	56.43	44.83	46.97	48.53	42.04	27.24	16.02	25.23	34.70	41.51	18.05	27.94
MVSNet [23]	43.48	55.99	28.55	25.07	50.79	53.96	50.86	47.90	34.69	-	-	-	-	-	-	-
R-MVSNet [24]	48.40	69.96	46.65	32.59	42.95	51.88	48.80	52.00	42.38	24.91	12.55	29.09	25.06	38.68	19.14	24.96
CasMVSNet [7]	56.42	76.36	58.45	46.20	55.53	56.11	54.02	58.17	46.56	31.12	19.81	38.46	29.10	43.87	27.36	28.11
PatchmatchNet [18]	53.15	66.99	52.64	43.24	54.87	52.87	49.54	54.21	50.81	32.31	23.69	37.73	30.04	41.80	28.31	32.29
EPP-MVSNet [10]	61.68	77.86	60.54	52.96	62.33	61.69	60.34	62.44	55.30	35.72	21.28	39.74	35.34	49.21	30.00	38.75
ATLAS-MVSNet [20]	60.71	77.62	61.96	49.55	61.63	60.04	58.69	63.58	52.59	-	-	-	-	-	-	-
DELS-MVS [17]	63.08	79.45	68.79	55.80	61.36	62.23	57.97	61.71	57.30	37.81	26.28	42.68	35.65	47.58	33.13	41.53
GeoMVSNet [27]	**65.89**	**81.64**	67.53	**55.78**	68.02	65.49	**67.19**	63.27	**58.22**	**41.52**	**30.23**	**46.53**	**39.98**	**53.05**	**35.98**	**43.34**
CER-MVS [11]	64.82	81.16	64.21	50.43	**70.73**	63.85	63.99	**65.90**	58.25	40.19	25.95	45.75	39.65	51.75	35.08	42.97
UniMVSNet [13]	64.36	81.20	66.43	53.11	63.46	**66.09**	64.84	62.23	57.53	38.96	28.33	44.36	39.74	52.89	33.80	34.63
GBiNet [12]	61.42	79.77	**67.69**	51.81	61.25	60.37	55.87	60.67	53.89	37.32	29.77	42.12	46.30	47.69	31.11	36.93
HAMMER (Ours)	61.70	78.45	59.25	54.33	62.80	63.20	59.57	61.72	54.23	36.13	24.17	40.07	38.14	49.56	31.54	33.31

Table 2. Results on the Tanks and Temples dataset of state-of-the-art MVS and our method. This table reflects the official Tanks and Temples benchmark where fusion parameters are free to be set individually for each scene. Nevertheless, we kept the parameters fixed for our method. Precision and recall is combined as *f-score* (higher is better). Best results are shown in bold and the runner-ups are underlined.

Method	**Mean**	**Fam**	**Fra**	**Hor**	**Lig**	**M60**	**Pan**	**Pla**	**Tra**	**Mean**	**Aud**	**Bal**	**Cou**	**Mus**	**Pal**	**Tem**
UniMVSNet [13]	60.39	79.28	**65.59**	41.67	**63.71**	61.58	58.77	60.33	52.18	28.05	13.77	29.95	26.70	46.12	28.40	23.36
GBiNet [12]	60.32	**79.29**	65.07	49.35	60.41	59.79	55.30	59.52	53.80	33.93	22.69	37.30	32.96	46.37	29.23	**35.03**
HAMMER (Ours)	**61.70**	78.45	59.25	**54.33**	62.80	**63.20**	**59.57**	**61.72**	**54.23**	**36.13**	**24.17**	**40.07**	**38.14**	**49.56**	**31.54**	33.31

Table 3. Results on the Tanks and Temples dataset when using the same fusion parameter setting for all scenes. Methods were chosen due to their performance, code availability and hardware requirements. We apply the average parameter setting that the corresponding authors suggested for the Tanks and Temples evaluation. Precision and recall is combined as *f-score* (higher is better).

Entropy threshold	**Acc.**	**Comp.**	**Overall**
baseline	0.400	0.235	0.318
no mask	0.390	**0.234**	0.312
1.3	0.384	0.239	0.311
0.8	0.334	0.262	**0.298**
0.7	0.326	0.270	**0.298**
0.6	0.320	0.280	0.300
0.1	**0.283**	0.447	0.365

Table 4. Ablations regarding entropy training and filtering on the DTU test dataset. The baseline is established from our network trained without entropy loss.

thresholds. By setting the threshold to the maximum entropy, which is equal to not applying any mask, we expected and confirmed a similar result to the baseline. However, it seems that just applying the loss in the training phase will already enhance the performance of the network.

Moreover, we found that using 5 instead of 4 feature stages can improve the result. Additional details are listed in the supplementary material.

6. Conclusion

We proposed HAMMER a deep neural network that is able to produce precise filter masks alongside accurate depth maps. Our novel entropy loss improves the depth map output of the network while also providing the required entropy map to filter out points before the fusion stage. HAMMER can be trained on an almost arbitrary depth resolution that is independent of GPU memory. Moreover, by using randomized matched patches in the training phase, we are able to train our network on 512x512 patches which leads to a very low memory requirement for training. Beside that, the main advantage of our method is that it does not require the adaption of fusion parameters to create a 3D model from its depth maps. We confirm the strong performance by evaluating with fixed fusion parameters on the very different benchmark scenes of DTU, Tanks and Temples intermediate and Tanks and Temples advanced.

Acknowledgement: This work has been supported by the FFG, Contract No. 881844: 'Pro^{2}Future'.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120(2):153–168, 2016. 1, 2, 3, 6

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2

[3] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2524–2534, 2020. 4

[4] Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–363. IEEE, 1996. 1

[5] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009. 1

[6] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Gipuma: Massively parallel multi-view stereo reconstruction. *Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e. V*, 25(361-369):1–2, 2016. 1, 2, 7

[7] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020. 1, 2, 3, 4, 7, 8

[8] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017. 4

[9] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 1, 2

[10] Xinjun Ma, Yue Gong, Qirui Wang, Jingwei Huang, Lei Chen, and Fan Yu. Epp-mvsnet: Epipolar-assembling based depth prediction for multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5732–5740, 2021. 7, 8

[11] Zeyu Ma, Zachary Teed, and Jia Deng. Multiview stereo with cascaded epipolar raft. In *European Conference on Computer Vision*, pages 734–750. Springer, 2022. 7, 8

[12] Zhenxing Mi, Chang Di, and Dan Xu. Generalized binary search network for highly-efficient multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12991–13000, 2022. 1, 2, 3, 4, 7, 8

[13] Rui Peng, Rongjie Wang, Zhenyu Wang, Yawen Lai, and Ronggang Wang. Rethinking depth estimation for multi-view stereo: A unified representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8645–8654, 2022. 1, 7, 8

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3, 4

[15] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 7, 8

[16] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 7, 8

[17] Christian Sormann, Emanuele Santellani, Mattia Rossi, Andreas Kuhn, and Friedrich Fraundorfer. Dels-mvs: Deep epipolar line search for multi-view stereo. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3087–3096, 2023. 7, 8

[18] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14194–14203, 2021. 1, 2, 7, 8

[19] Zizhuang Wei, Qingtian Zhu, Chen Min, Yisong Chen, and Guoping Wang. Aa-rmvsnet: Adaptive aggregation recurrent multi-view stereo network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6187–6196, 2021. 1

[20] Rafael Weilharter and Friedrich Fraundorfer. Atlas-mvsnet: Attention layers for feature extraction and cost volume regularization in multi-view stereo. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3557–3563. IEEE, 2022. 3, 7, 8

[21] Jianfeng Yan, Zizhuang Wei, Hongwei Yi, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Dense hybrid recurrent multi-view stereo net with dynamic consistency checking. In *European Conference on Computer Vision*, pages 674–689. Springer, 2020. 2

[22] Jiayu Yang, Wei Mao, Jose M Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4877–4886, 2020. 4

[23] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018. 1, 2, 6, 7, 8

[24] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5525–5534, 2019. 1, 2, 4, 7, 8

[25] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo net-

works. *Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 6

[26] Jingyang Zhang, Yao Yao, Shiwei Li, Zixin Luo, and Tian Fang. Visibility-aware multi-view stereo network. *arXiv preprint arXiv:2008.07928*, 2020. 2

[27] Zhe Zhang, Rui Peng, Yuxi Hu, and Ronggang Wang. Geomvsnet: Learning multi-view stereo with geometry perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21508–21518, 2023. 1, 7, 8