

# Learning Non-volumetric Depth Fusion using Successive Reprojections

Simon Donné      Andreas Geiger

Autonomous Vision Group

MPI for Intelligent Systems and University of Tübingen

{simon.donne, andreas.geiger}@tue.mpg.de

## Abstract

Given a set of input views, multi-view stereopsis techniques estimate depth maps to represent the 3D reconstruction of the scene; these are fused into a single, consistent, reconstruction – most often a point cloud. In this work we propose to learn an auto-regressive depth refinement directly from data. While deep learning has improved the accuracy and speed of depth estimation significantly, learned MVS techniques remain limited to the planesweeping paradigm. We refine a set of input depth maps by successively reprojecting information from neighbouring views to leverage multi-view constraints. Compared to learning-based volumetric fusion techniques, an image-based representation allows significantly more detailed reconstructions; compared to traditional point-based techniques, our method learns noise suppression and surface completion in a data-driven fashion. Due to the limited availability of high-quality reconstruction datasets with ground truth, we introduce two novel synthetic datasets to (pre-)train our network. Our approach is able to improve both the output depth maps and the reconstructed point cloud, for both learned and traditional depth estimation front-ends, on both synthetic and real data.

## 1. Introduction

Multi-view stereopsis techniques constitute the current state-of-the-art in 3D point cloud reconstruction [21]. Given a set of images and camera matrices, MVS techniques estimate depth maps for all input views and subsequently merge them into a consistent 3D reconstruction. While the deep learning paradigm has led to drastic improvements in the depth estimation step itself, the existing learned MVS approaches [11, 38] consist of plane-sweeping, followed by classical depth fusion approaches [7, 30] (which mainly filter out invalid estimates). Instead, we propose to learn depth map fusion from data; by incorporating and fusing information from neighbouring views we refine the depth map estimate of a central view. We dub our approach DeFuSR: Depth Fusion Through Successive Reprojections.

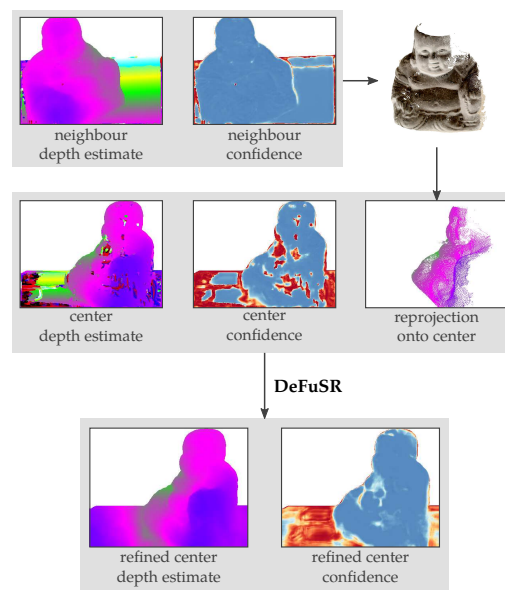


Figure 1: The center view depth estimate is inaccurate around the center of the Buddha, even though the neighbouring view has a confident estimate for these areas. By reprojecting the neighbour’s information onto the center image, we efficiently encode this information for the refinement network to resolve the uncertain areas. Iteratively performing this refinement further improves the estimates.

In volumetric space, learned approaches [2, 16, 17, 29] that fuse depth information from multiple views have shown great promise but are inherently limited because of computation time and memory requirements. Working in the image domain bypasses these scaling issues [8, 30, 38], but existing image-based fusing techniques focus on filtering out bad estimates in the fusion step rather than improving them. However, neighbouring views often contain information that is missing in the current view, as illustrated in Figure 1. We show that there is still a significant margin for improvement of the depth estimates by auto-regressively incorporating information from neighbouring views.

As the absence of large-scale high-quality ground truth depth maps is a potential hurdle in training our model, we introduce two novel synthetic MVS datasets for pre-training. The first is similar to Flying Chairs [4] and Flying Things 3D [24]. To close the domain gap between this dataset and the DTU [14] dataset, we also use Unreal Engine to render the unrealDTU dataset, a drop-in replacement for DTU.

To summarize our contributions: we propose autoregressive **learned depth fusion in the image domain**, we create **two synthetic MVS datasets** for pretraining, we empirically motivate the design choices with **ablation studies**, and we compare our approach with **state-of-the-art depth fusion baselines**. Code and datasets are available via <https://github.com/simon-donne/defusr/>.

## 2. Related Work

We first discuss the estimation of the depth maps before we go on to discuss their fusion. We give only an overview of the most influential recent work in both areas; an exhaustive historical overview can be found in [6].

### 2.1. Multi-view Depth Estimation

**Traditional MVS:** Based on the popular PatchMatch algorithm for rectified stereomatching [1], Gipuma [8] and COLMAP [30] are popular state-of-the-art approaches [21, 31]; they propagate best fitting planes to estimate per-pixel depth. While Gipuma selects neighbouring views on a per-view basis, COLMAP does so per-pixel for better results.

**Deep Stereo:** Initial learning-based depth estimation considered only the binocular task, using Siamese Networks to learn patch-based descriptors that are aggregated using winner-takes all [40] or global optimization [41]. By combining patch description, matching cost and cost volume processing in a single network, disparity estimation can be learned end-to-end [19, 20, 23]. Finally, Ummenhofer et al. [35] demonstrate a model which jointly predicts depth, camera motion, normals and optical flow from two views.

**Deep MVS:** Hartmann et al. successfully showed the generalization of two-view matching-based costs to multiple views [10]. While the end-to-end approaches for disparity estimation mentioned above were restricted to the binocular case, Leroy et al. [22], DeepMVS [11] and MVSNet [38] show that depth map prediction can benefit from multiple input views. Similarly, Xu et al. recently proposed AMHMVS [36], a learning-based version of Gipuma [8]. Paschalidou et al. [27] exploit the combination of deep learning and Markov random fields for highly accurate depth maps, but are restricted to relatively low resolutions. All of these methods have focused on the depth estimation problem. However, we show that fusing and incorporating depth from multiple views is a viable avenue for improvements.

### 2.2. Depth Map Fusion

Depth-based stereopsis techniques are subsequently faced with the task of fusing a set of depth maps into a consistent reconstruction. This part, too, can be split up into volumetric and image-based fusion approaches. Intuitively, volumetric fusion can better leverage spatial information, but image-domain techniques such as ours are more efficient and lightweight, enabling higher output resolutions.

**Volumetric Fusion**, initially proposed by Curless et al. [3], was made increasingly popular by Zach et al. [39] and KinectFusion [26], fusing various depth maps into a single truncated signed distance field (TSDF). Leroy et al. have recently integrated this TSDF-based fusion into an end-to-end pipeline [22]. Fuhrmann et al. discuss how to handle the case of varying viewing distances and at the same time do away with the volumetric grid in favour of a point list [5] which scales better. Other non-learning-based techniques have been proposed to counter this scaling behaviour, such as a hybrid Delaunay-volumetric approach [25] and octrees [34]. The first does not lean itself well for learning-based approaches, but three concurrent works have leveraged hierarchical surface representations (i.e., octrees) to improve execution speed [9, 29, 32]. However, even such approaches have issues scaling beyond  $512^3$  voxels: eventually, they hit a computational ceiling. By working in the image domain, we largely sidestep the scaling issue and can additionally lean on the large amount of work and understanding available for image-based deep learning.

**Image-based Fusion** promises quadratic rather than cubic scaling. Traditionally, it only discards reconstructed points that are not supported by multiple views. This is implemented in Gipuma as the Fusibile algorithm [8], and Xu et al. [36] use the same fusion technique in AMHMVS, their learning-based version of Gipuma. In COLMAP [30], the accepted pixels are clustered in “consistent pixel clusters” that are combined into a single reconstructed point cloud: clusters not supported by a minimum number of views are discarded. Similarly, Poggi et al. [28] and Tosi et al. [33] have leveraged deep learning to yield confidence estimates.

While the former techniques filter out bad depth estimates, they do not attempt to improve the estimates. We argue that the depth maps can still be significantly improved by incorporating information from neighbouring views. To the best of our knowledge, learning-based refinement of depth maps was only done in a single-view setting [15, 37].

We aim to learn depth map fusion and refinement from a variable number of input views – the zero-neighbour variant of our approach serves as the single-image baseline similar to [15] and [37]. Our combined approach notably improves the quality of the fused point clouds, quantified in terms of the Chamfer distance (see Section 5), and at the same time yields improved depth maps for all input views.

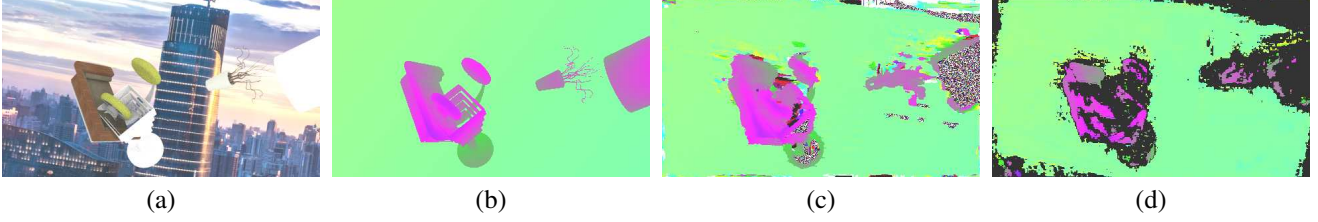


Figure 2: Example from our synthetic dataset: an input image (a) with the corresponding ground truth depth map (b). The depth map estimates from COLMAP (c) and MVSNet (d) show the issues in poorly constrained areas, usually because of occlusions and homogeneous areas. While MVSNet also returns a confidence estimate for its estimate, we bootstrap our method with single-view confidence estimation in the case of COLMAP inputs.



Figure 3: Examples from our unrealDTU dataset: similar in set-up to DTU, we observe a series of objects on a table from a set of cameras scattered across one octant of a sphere.

### 3. Datasets

For evaluation and training, we consider the DTU MVS dataset [14]. Unfortunately, DTU lacks perfect ground truth: a potential hurdle for the learning task. To tackle this, we have constructed two new synthetic datasets for pre-training; see the supplementary material for more details.

The first, as seen in Figure 2, is similar to Flying Chairs [4] and Flying Things 3D [24]. We create ten observations of a static scene rather than only two views of a non-rigid scene. Rendered with Blender, each scene consists of 10-20 ShapeNet objects randomly placed in front of a slanted background plane with an arbitrary image texture.

Secondly, we also introduce a more realistic dataset to close the domain gap between the above dataset and real imagery. This second dataset is a drop-in replacement for the DTU dataset, rendered within Unreal Engine (see Figure 3), sporting perfect ground truth and more realistic rendering.

### 4. Method

We now outline the various aspects of our approach. We assume a set of depth map estimates as input, in this work from one of two front-ends: the traditional COLMAP [30] or the learning-based MVSNet [38]. The photometric depth map estimates from COLMAP are extremely noisy in poorly constrained areas (see Figure 2); although this would interfere with our reprojection step (see later), it proves to be straightforward to filter such areas out. We bootstrap our method by estimating the confidence for the input estimates – we discuss this in more detail at the end of this section.

#### 4.1. Network overview

Our network is summarized in Figure 4. We first outline the entire process before discussing each aspect in detail. For an exhaustive listing, we refer to the supplementary.

As mentioned before, the depth fusion step happens entirely in the image domain. To encode the information from neighbouring views, we project their depth maps and image features (obtained from a image-local network) onto the center view. After pooling the information of all neighbours, we have a two-sided approach: one head to residually refine the input depth values (with limited spatial support), and another head that is intended to inpaint large unknown areas (with much larger spatial support). A third network weights the two options to yield the output estimate. A final network predicts the confidence of the refined estimate. We do not use any normalization layers in the refinement parts of the network, as absolute depth values need to be preserved.

#### Neighbour reprojection

Consider the set of  $N$  images  $I_n(\mathbf{u})$  with corresponding camera matrices  $P_n = \mathbf{K}_n [\mathbf{R}_n | \mathbf{t}_n]$ , and estimated depth values  $d_n(\mathbf{u})$  for each input pixel  $\mathbf{u} = [u, v, 1]^T$ . The 3D point corresponding with a given pixel is then given by

$$\mathbf{x}_n(\mathbf{u}) = \mathbf{R}_n^T \mathbf{K}_n^{-1} (d_n(\mathbf{u}) \mathbf{u} - \mathbf{K}_n \mathbf{t}_n). \quad (1)$$

Those  $\mathbf{x}_n(\mathbf{u})$  for which the input confidence is larger than 0.5 are then projected onto the center view 0. Call  $\mathbf{u}_{n \rightarrow m}(\mathbf{u}) = P_m \mathbf{x}_n(\mathbf{u})$  the projection of  $\mathbf{x}_n(\mathbf{u})$  onto neighbour  $m$ , and  $z_{n \rightarrow m}(\mathbf{u}) = [0 \ 0 \ 1] \mathbf{u}_{n \rightarrow m}(\mathbf{u})$  its depth.

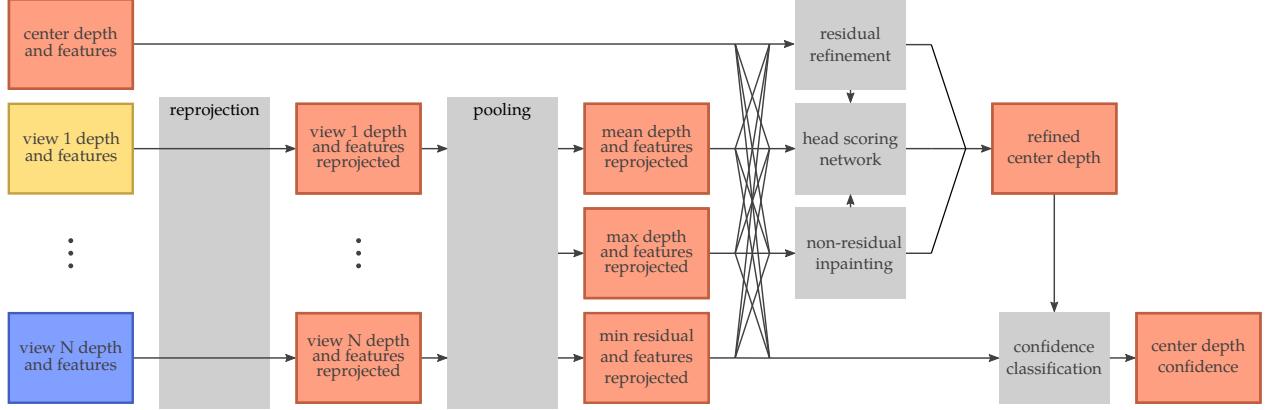


Figure 4: Overview of our proposed fusion network. A difference in coloring represents a difference in vantage point, i.e. the information is represented in different image planes. As outlined in Section 4.1, neighbouring views are first reprojected, and then passed alongside the center depth estimate and the observed image. The output of the network is an improved version of the input depth of the reference view as well as a confidence map for this output.

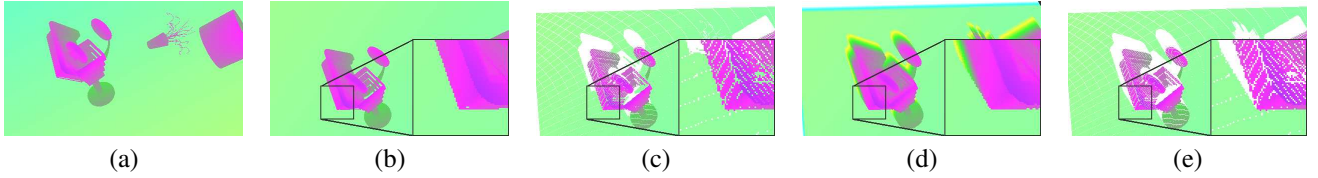


Figure 5: Example of the depth reprojection, bound calculation, and culling results. A neighbour image (a) is reprojected onto the center view (b), resulting in unfiltered reprojection (c). Calculating the lower bound as in Section 4.1 yields (d). Finally, we filter (c) with (d) as outlined in Section 4.1 to result in the culled depth map (e). Note in the crop-outs how the bound calculation completes depth edges, while the culling step removes bleed-through.

The z-buffer in view 0 based on neighbour  $n$  is then

$$z_n(\mathbf{u}) = \min_{\mathbf{u}_n \in \Omega_n(\mathbf{u})} z_{n \rightarrow 0}(\mathbf{u}_n), \quad (2)$$

where  $\Omega_n(\mathbf{u}) = \{\mathbf{u} \in \Omega | \mathbf{P}_0 \mathbf{x}_n(\mathbf{u}_n) \sim \mathbf{u}\}$  is the collection of pixels in view  $n$  that project onto  $\mathbf{u}$  in view 0.

We call  $\mathbf{u}_n(\mathbf{u})$  the pixel in view  $n$  for which  $z_n(\mathbf{u}) = z_{n \rightarrow m}(\mathbf{u}_n(\mathbf{u}))$ , i.e. the pixel responsible for the entry in the z-buffer. We now construct the reprojected image as

$$\tilde{\mathbf{I}}_n(\mathbf{u}) = \mathbf{I}_n(\mathbf{u}_n(\mathbf{u})). \quad (3)$$

Note the apparent similarity with Spatial Transformer Networks [13], at least as far as the projection of the image features goes; the geometry involved in the reprojection of the depth maps is not captured by a Spatial Transformer.

The reprojected depth and features have two major issues (as a result of the aliasing inherent in the reprojection, which is essentially a form of resampling). First of all, sharp depth edges in the neighbour view are often smeared over a large area in the center view, due to the difference in vantage points. While they do not constitute a evidence-supported surface, they do imply evidence-supported free space which

contains information valuable to the network; we encode this into a lower bound image. Secondly, because of the aliasing in the reprojection step, background surfaces bleed through foreground surfaces. We now detail how to address both issues, as visualized in Figure 5.

**Minimum depth bound** To encode the free space implied by a neighbour’s depth estimate, we calculate a lower bound depth map  $b_n(\mathbf{u})$  for the center view with respect to neighbour  $n$  as the lowest depth hypotheses along each pixel’s ray that is no longer observed as free space by the neighbour image (see Figure 6). As far as the reference view is concerned, this is the lower bound on the depth of that pixel as implied by that neighbour’s depth map estimate. In terms of the previous notations, we can express the lower bound  $g_n(\mathbf{u})$  from neighbour  $n$  as

$$g_n(\mathbf{u}) = \min \{d > 0 \mid d_m(\mathbf{u}_{0 \rightarrow n}(\mathbf{u})) > z_{0 \rightarrow n}(\mathbf{u})\}. \quad (4)$$

**Culling invisible surfaces** We now cull invisible surfaces in the reprojected depth  $z_n(\mathbf{u})$ : any pixel significantly beyond the lower bound  $b_n(\mathbf{u})$  is considered to be invisible to the center view and is discarded in the culled version  $\tilde{z}_n(\mathbf{u})$ . The threshold we use here is  $1/1000^{\text{th}}$  of the maximum depth in the scene (determined experimentally).



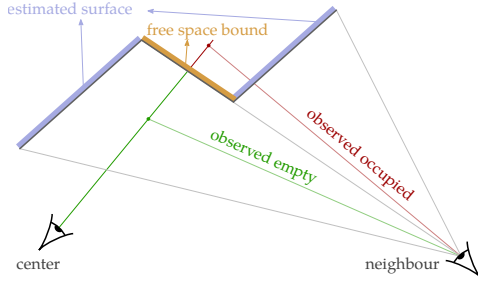


Figure 6: Visualization of the lower bound calculation. For each pixel in the center view, we find the lowest depth value for which the unprojected point is no longer viewed as empty space by the neighbouring camera.

**Value of initial confidence estimation** MVSNet yields a confidence for its estimates, but for COLMAP, we bootstrap our method with a confidence estimation for the input depth. Figure 7 illustrates the necessity of this confidence filtering: the bounds image and culled reprojection become noticeably cleaner. This confidence estimation is described below.

### Neighbour pooling

We perform three types of per-pixel pooling over the reprojected neighbour information. First of all, we calculate the mean and maximum depth bound and culled reprojected depth, as well as the average reprojected feature and the feature corresponding to the maximum culled reprojected depth. We also extract the culled reprojected depth that is closest to the center view estimate, and its feature. These are passed to the refinement and classification heads, along with the input depth estimate and image features.

### Depth refinement

The depth refinement step consists of two steps. In a first step, the center view depth estimate and features, as well as the pooled reprojected neighbour information (we will refer to these as the “shared features”), are processed by two networks: a local residual depth refinement module (a UNet block with depth 1 whose output is added to the input depth estimate) and a depth inpainting module (a UNet block with depth 5). Finally, a scoring network takes the output of the two other heads as input in addition to the shared features and outputs scores for both the residual refinement and inpainting alternatives. These are softmaxed and used to weight both outputs.

### Confidence classification

Finally, the last part of the network takes both the shared features and the final depth estimate as input to yield a confidence classification of the output depth. This network is a UNet with depth 4, outputting a single channel on which a sigmoid is applied to acquire the final confidence prediction.

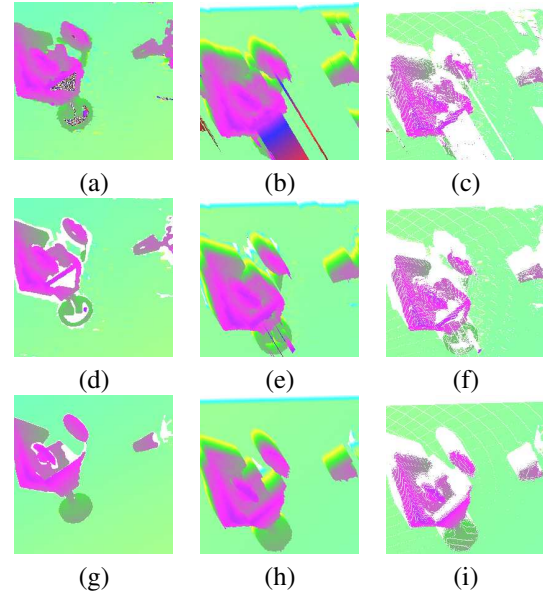


Figure 7: The necessity of the initial confidence estimate for the depth map inputs. While the noisy input depth map (a) yields noticeable artifacts in the bounds image (b) and culled reprojected image (c), the confidence-filtered estimate (d) yields cleaner bounds (e) and culled reprojection (f). The final row shows the same steps after one iteration of refinement: the refined neighbour estimate (g), and its implied bounds (h) and reprojection (i).

### Training

As the feature reprojection is differentiable in terms of the features themselves (but not in terms of the depth values), we train the entire architecture end-to-end. However, training the different refinement/inpainting/head scoring networks is challenging and training the confidence network from the get-go causes it to degenerate into estimating zero confidence everywhere (which is correct in the initial epochs).

To mitigate this, we apply curriculum learning. In an initial phase, only the inpainting head is used. After a while, we enable the weighting network between the two heads, but keep the residual refinement disabled. Once the classification network makes valid choices between inpainted and input depth, we enable the residual refinement and the confidence classification: utilizing the entirety of our architecture.

The refined depth output is supervised with an  $L1$  loss on both the depth values and their gradients. The confidence is supervised with a binary cross-entropy loss, where pixels are assumed to be trusted if they are within a given threshold of the ground truth value. The confidence loss is only back-propagated through the confidence classification block to prevent it from degenerating the depth estimate to make its own optimization easier; as a result we require no weighting between both losses as they affect distinct sets of weights.



Figure 8: The depth maps used for supervision on the DTU dataset. For a given input image (a), the dataset contains a reference point cloud (b) obtained with a structured light scanner. After creating a watertight mesh (c) from this point cloud and projecting that, we reject any point in the point cloud which gets projected behind this surface from the supervision depth map (d). Additionally, we reject the points of the white table.

To unify the world scales of the different datasets, we scale all scenes such that the used depth range is roughly between 1 and 5. To reduce the sensitivity to this scale factor, as well as augment the training set, we scale each minibatch element by a random value between 0.5 and 2.0. After inference, we undo this scaling.

### Supervision

Our approach requires ground truth depth maps for supervision. In the synthetic case, flawless ground truth depth is provided by the rendering engine. For the DTU dataset, however, ground truth consists of point clouds reconstructed by a structured light scanner [14]. Creating depth maps out of these point clouds, as before, faces the issue of bleedthrough as in Figure 5c. To address this issue, we perform Poisson surface reconstruction of the point cloud to yield a watertight mesh [18]; any point in the point cloud which projects behind this surface is rejected from the ground truth depth maps. While this method works well, it was not viable for use inside the network because of its relatively low speed. Finally, we also reject points on the surface of the white table – this cannot be reconstructed by photometric methods and our network should not be penalized for incorrect depth estimates here: instead, we supervise those areas with zero confidence. These issues are illustrated in Figure 8.

## 5. Experimental Evaluation

In what follows we empirically show the benefit of our approach. To quantify performance, we recall the accuracy and completeness metrics from the DTU dataset [14]:

$$\text{accuracy}(\mathbf{u}, n) = \min_g \|\mathbf{x}_g - \mathbf{x}_n(\mathbf{u})\|_2, \text{ and} \quad (5)$$

$$\text{completeness}(g) = \min_{\mathbf{u} \in \Omega, n} \|\mathbf{x}_g - \mathbf{x}_n(\mathbf{u})\|_2. \quad (6)$$

Accuracy indicates how close a reconstructed point lies to the ground truth, while completeness expresses how close a reference point lies to our reconstruction (for both metrics, lower is better); the Chamfer distance is defined as the algebraic mean of accuracy and completeness.

We are mostly interested in the percentage of points whose accuracy or completeness is lower than  $\tau = 2.0\text{mm}$ , which we consider more indicative than the average accuracy or completeness metrics: it matters little whether an erroneous surface is 10 or 20 cm offset – yet this strongly affects the global averages. We report on both per-view, to quantify individual depth map quality, and for the final point cloud. More results, including results on the new synthetic datasets for pre-training and the absolute distances for DTU, are provided in the supplementary.

To create the fused point cloud for our network, we simultaneously thin the point clouds and remove outliers (similar to Fusibile). The initial point cloud from the output of our network is given by all pixels for which the predicted confidence is larger than a given threshold – the choice for this threshold is empirically decided below. For each point, we count the reconstructed points within a given threshold  $\tau$ :

$$c_\tau(\mathbf{u}, n) = \sum_{\mathbf{u}_2 \in \Omega, n_2} I(\|\mathbf{x}_n(\mathbf{u}) - \mathbf{x}_{n_2}(\mathbf{u}_2)\|_2 < \tau), \quad (7)$$

where  $I(\cdot)$  is the indicator function.  $\mathbf{x}_n(\mathbf{u})$  is accepted in the final cloud with probability  $I(c_\tau(\mathbf{u}, n) > 1)/c_{\tau/5}(\mathbf{u}, n)$ . Obvious outliers, with no points closer than  $\tau$ , are rejected. For the other points, rejection probability is inversely proportional to the number of points closer than  $\tau/5$ , mitigating the effect of non-uniform density of the reconstruction.

All evaluations are at a resolution of  $480 \times 270$ , for feasibility and because MVSNet is also restricted to this.

### 5.1. Selecting the confidence thresholds

The confidence classification of our approach is binary; whether or not a depth estimate lies within a given threshold  $\tau_d$  of the ground truth. Only depth estimates for which this predicted probability is above  $\tau_{\text{prob}}$  are considered for the final point cloud. Figure 9 illustrates that training the confidence classification for various  $\tau_d$  yield the same trade-off between accuracy and completeness percentages. Based on these curves, we select  $\tau_d = 2$  and  $\tau_{\text{prob}} = 0.5$  for the evaluation to maximize the sums of both percentages.

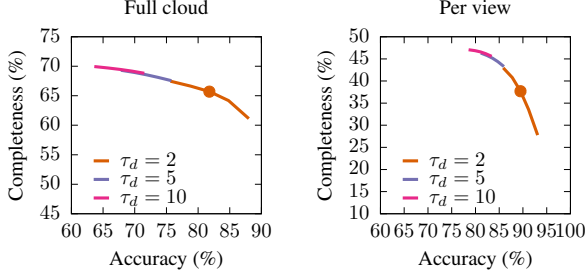


Figure 9: The percentage of points with accuracy respectively completeness below 2.0, over the DTU validation set, for varying values of  $\tau_d$ . The curves result from varying  $\tau_{\text{prob}}$ ; note that the evaluated options essentially lead to a continuation of the same curve. We select  $\tau_d = 2$  and  $\tau_{\text{prob}} = 0.5$  as the values that optimize the sum of both percentages.

## 5.2. Selecting and leveraging neighbours

We consider three strategies for neighbour selection: selecting the closest views, selecting the most far-away views, and selecting a mix of both. We evaluate three separate networks with these strategies to perform refinement of COLMAP depth estimates with twelve neighbouring views. Table 1 shows the performance of these strategies, after three iterations of refinement, compared to the result of COLMAP’s fusion. The *mixed* strategy proved to be the most efficient: while far-away views clearly contain valuable information, close-by neighbours should not be neglected. In following experiments, we have always used the *mixed* strategy. While there is a practical limit (we have limited ourselves to twelve neighbours in this work), Table 2 shows that, as expected, using more neighbours leads to better results.

## 5.3. Refining MVSNet estimates

Finally, we evaluate the use of our network architecture for refining the MVSNet depth estimates. As shown in Table 3, our approach is not able to refine the MVSNet estimates much; while the per-view accuracy increases noticeably, the other metrics remain roughly the same.

The raw MVSNet estimates perform better than the COLMAP depth estimates. Refining the COLMAP estimates with our approach, however, significantly improves over the MVSNet results (refined or otherwise). We observe (e.g. Figure 10) that the COLMAP estimates are much more view-point dependent: surface patches observed by many neighbours are reconstructed more accurately than others. MVSNet, as a learned technique, was trained to optimize the L1 error and appears to smooth these errors out over the surface. Intuitively, the former case indeed allows for more improvements by our approach, by propagating these accurate areas to neighbouring views.

Table 1: Quantitative evaluation of the neighbour selection. Using twelve neighbours for three iterations of refinement, leveraging information from both close-by and far-away neighbours yields the best results, mostly improving completeness compared to the COLMAP fusion result.

		COLMAP	nearest	mixed	furthest
per-view	acc. (%)	66	91	89	86
	comp. (%)	40	38	45	37
	mean (%)	52	64	<b>67</b>	62
full	acc. (%)	73	83	80	74
	comp. (%)	72	72	84	76
	mean (%)	72	78	<b>82</b>	75

Table 2: Quantitative evaluation of the number of neighbours. Using zero, four, or twelve neighbours for three iterations of refinement. As expected, more neighbours results in better performance, and too few neighbours performs worse than COLMAP’s fusion approach (which essentially uses all 48).

		COLMAP	neighbours		
			0	4	12
per-view	acc. (%)	66	91	92	89
	comp. (%)	40	28	31	45
	mean (%)	52	59	62	<b>67</b>
full	acc. (%)	73	81	84	80
	comp. (%)	72	66	64	84
	mean (%)	72	74	74	<b>82</b>

Table 3: Refining the MVSNet output depth estimates using three iterations of our approach. The per-view accuracy noticeably increases, while other metrics see a slight drop.

		MVSNet	Refined (it. 3)
per-view	acc. (%)	76	92
	comp. (%)	35	34
	mean (%)	55	<b>63</b>
full	acc. (%)	88	86
	comp. (%)	66	65
	mean (%)	<b>77</b>	76

## 5.4. Qualitative results

Figure 10 illustrates that performing more than one iteration is paramount to a good result but as the gains level out quickly, we have settled on three refinement steps.

Neighbouring views are crucial to filling in missing areas of individual views, as Figure 11 illustrates: here, the entire right side of the box is missing in the input estimate. Single-view refinement is not able to fill in this missing area and does not benefit from multiple refinement iterations. Refinement based on 12 neighbours, however, propagates information from other views and further improves the estimate over the next iteration, leveraging the now-improved information in those neighbours.

Having focused on structure throughout this work, rather than appearance, our reconstructed point clouds look more noisy due to lighting changes (see Figure 12), while COLMAP fusion averages colour over the input views.



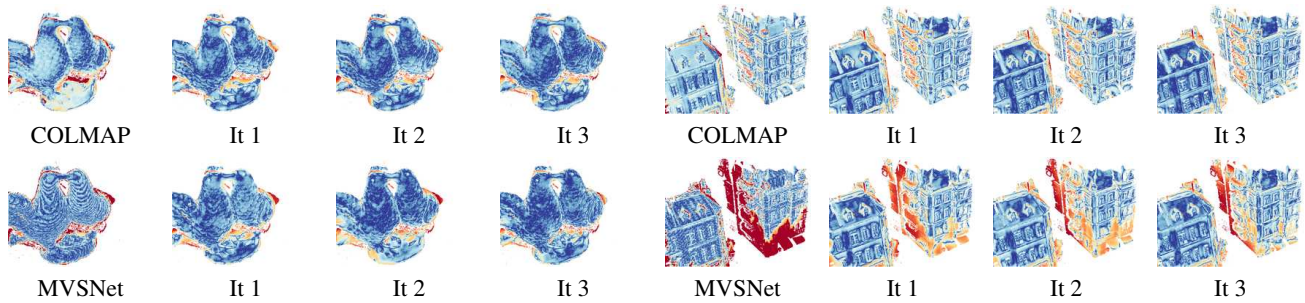


Figure 10: Visualization of the depth map error over multiple iterations, for two elements of the DTU test set (darker blue is better). The first two iterations are the most significant, after that the improvements level out. Object elements not available in any view (the middle part of the buildings on the bottom right) cannot be recovered.

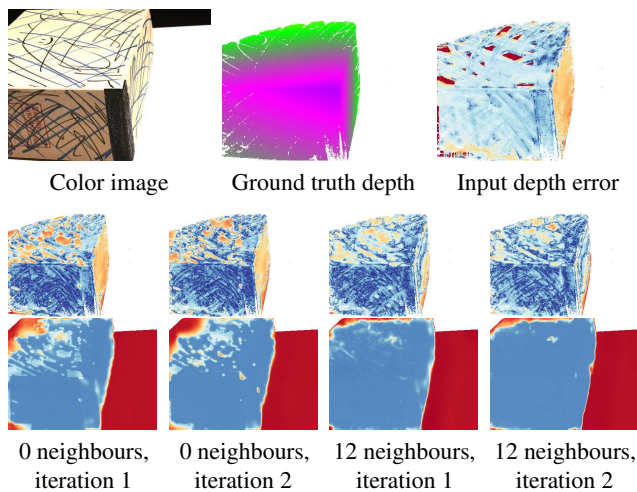


Figure 11: Output depth errors (middle) and confidence (bottom) for our approach (darker blue is better). Without leveraging neighbouring views, additional iterations yield little benefit. Neighbour information leads to better depth estimates and confidence, further improving over iterations.

Finally, we also provided a qualitative comparison on a small handheld capture. Without retraining the network, we process the COLMAP depth estimates resulting from 12 images taken with a smartphone, both for a DTU-style object on a table and a less familiar setting of a large couch. While the confidence network trained on DTU has issues around depth discontinuities, Figure 13 shows that the surfaces are well reconstructed.

## 6. Conclusion and Future Work

We have introduced a novel learning-based approach to depth fusion, DeFuSR: by iteratively propagating information from neighbouring views, we refine the input depth maps. We have shown the importance of both neighbourhood information and successive refinement for this problem, resulting in significantly more accurate and complete per-view and overall reconstructions.

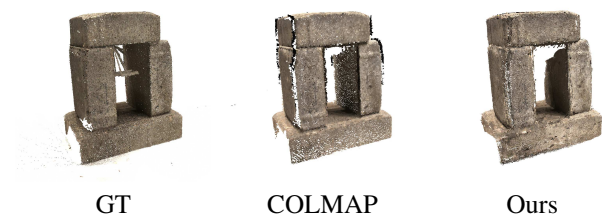


Figure 12: Example reconstructed cloud for an element of the test set. Note the significant imperfections in the reference point cloud (left). A visual drawback of our approach is that we focus solely on structure; the fusion step used by COLMAP innately also averages out the difference in color appearance between various views.



Figure 13: Two scenes captured with a smartphone (12 images). Note that the surfaces are well estimated (the green pillow and the gnome's nose) and holes are completed correctly (the black pillow, the gnome's hat).

We mention two directions for future work. First of all, our training loss is the L1 loss, which is known to have a tendency towards smooth outputs; alternative loss functions, such as the PatchGAN [12] can help mitigate this. Secondly, we have selected neighbours on the image level. Ideally, the selection of neighbours would happen more finegrained, and integrated into the learning pipeline, e.g. in the form of attention networks.



## References

- [1] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo - stereo matching with slanted support windows. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2011. 2
- [2] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1
- [3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *ACM Trans. on Graphics*, 1996. 2
- [4] A. Dosovitskiy, P. Fischer, E. Ilg, P. Haeusser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015. 2, 3
- [5] S. Fuhrmann and M. Goesele. Floating scale surface reconstruction. *TG*, 2014. 2
- [6] Y. Furukawa, C. Hernández, and al. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2015. 2
- [7] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015. 1
- [8] S. Galliani, K. Lasinger, and K. Schindler. Gipuma: Massively parallel multi-view stereo reconstruction. *Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e. V.*, 25:361–369, 2016. 1, 2
- [9] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. *arXiv.org*, 1704.00710, 2017. 2
- [10] W. Hartmann, S. Galliani, M. Havlena, L. Van Gool, and K. Schindler. Learned multi-patch similarity. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 2
- [11] P. Huang, K. Matzen, J. Kopf, N. Ahuja, and J. Huang. Deepmvs: Learning multi-view stereopsis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2
- [12] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 8
- [13] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 4
- [14] R. R. Jensen, A. L. Dahl, G. Vogiatzis, E. Tola, and H. Aanaes. Large scale multi-view stereopsis evaluation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2, 3, 6
- [15] J. Jeon and S. Lee. Reconstruction-based pairwise depth dataset for depth image enhancement using cnn. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [16] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang. SurfaceNet: an end-to-end 3d neural network for multiview stereopsis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1
- [17] A. Kar, C. Häne, and J. Malik. Learning a multi-view stereo machine. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1
- [18] M. M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics*, 32(3):29, 2013. 6
- [19] A. Kendall, H. Martirosyan, S. Dasgupta, and P. Henry. End-to-end learning of geometry and context for deep stereo regression. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 2
- [20] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. *arXiv.org*, 2018. 2
- [21] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. on Graphics*, 36(4), 2017. 1, 2
- [22] V. Leroy, J.-S. Franco, and E. Boyer. Shape reconstruction using volume sweeping and learned photoconsistency. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [23] W. Luo, A. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2
- [24] N. Mayer, E. Ilg, P. Haeusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3
- [25] C. Mostegel, R. Prettenhaler, F. Fraundorfer, and H. Bischof. Scalable surface reconstruction from point clouds with extreme scale and density diversity. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2
- [26] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 2
- [27] D. Paschalidou, A. O. Ulusoy, C. Schmitt, L. van Gool, and A. Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [28] M. Poggi and S. Mattoccia. Learning from scratch a confidence measure. In *BMVC*, 2016. 2
- [29] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger. OctNetFusion: Learning depth fusion from data. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017. 1, 2
- [30] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1, 2, 3
- [31] T. Schöps, J. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

- [32] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 2
- [33] F. Tosi, M. Poggi, A. Benincasa, and S. Mattoccia. Beyond local reasoning for stereo confidence estimation with deep learning. In *ECCV*, 2018. 2
- [34] B. Ummenhofer and T. Brox. Global, dense multiscale reconstruction for a billion points. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015. 2
- [35] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2
- [36] Q. Xu and W. Tao. Multi-view stereo with asymmetric checkerboard propagation and multi-hypothesis joint view selection. *arXiv.org*, 2018. 2
- [37] S. Yan, C. Wu, L. Wang, F. Xu, L. An, K. Guo, and Y. Liu. Ddrnet: Depth map denoising and refinement for consumer depth cameras using cascaded cnns. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2
- [38] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *arXiv.org*, abs/1804.02505, 2018. 1, 2, 3
- [39] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2007. 2
- [40] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. 2015. 2
- [41] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research (JMLR)*, 17(65):1–32, 2016. 2