Supplementary Material for
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@tue.mpg.de, andreas.geiger@tue.mpg.de

Abstract

In this supplementary document, we first give an exhaustive overview of the network architecture in Section 1, with a brief discussion on runtime. More qualitative and quantitative results, both on depth maps and point clouds, are given in Section 2. The supplementary video visualizes the concepts of reprojection and bounding/culling the reprojections, as well as the effect of the iterative refinement on the final point cloud.

1. Network architecture

We reprise the schematic overview of our network approach in Figure 1. As introduced in the main text, the depth fusion step happens entirely in the image domain. In order to encode the information from neighbouring views, their depth maps and image features are cast into space and reprojected onto the center view. After pooling the information from all observed neighbours, our approach comprises several streams. The first residually refines the input depth values to improve nearly-correct depth estimates, having only a limited spatial support. A second stream is intended to inpaint large unknown areas (with a much larger spatial support). A third network stream predicts the weighting between these two options to yield the output estimate. Finally, a confidence network predicts the confidence in this final estimate: the probability that the estimate is within a certain threshold of the ground truth. In order to preserve the absolute depth values, we do not use any normalization layers in the refinement streams of the network.

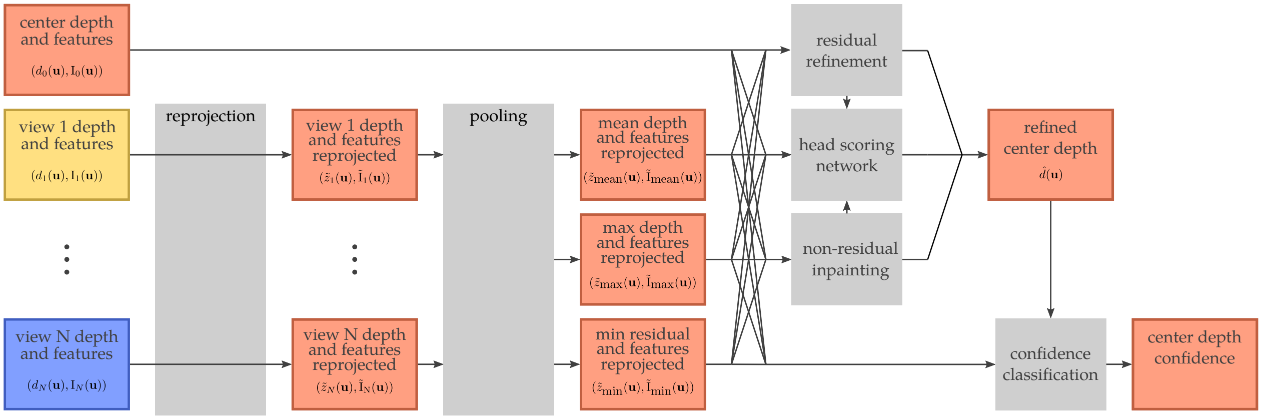
center depth and features
$(d_0(\mathbf{u}), \mathrm{I}_0(\mathbf{u}))$
view 1 depth and features
$(d_1(\mathbf{u}), \mathrm{I}_1(\mathbf{u}))$
view N depth and features
$(d_N(\mathbf{u}), \mathrm{I}_N(\mathbf{u}))$
reprojection
view 1 depth and features reprojected
$(\tilde{z}_1(\mathbf{u}), \tilde{\mathrm{I}}_1(\mathbf{u}))$
view N depth and features reprojected
$(\tilde{z}_N(\mathbf{u}), \tilde{\mathrm{I}}_\mathrm{N}(\mathbf{u}))$
pooling
mean depth and features reprojected
$(\tilde{z}_\mathrm{mean}(\mathbf{u}), \tilde{\mathrm{I}}_\mathrm{mean}(\mathbf{u}))$
max depth and features reprojected
$(\tilde{z}_\mathrm{max}(\mathbf{u}), \tilde{\mathrm{I}}_\mathrm{max}(\mathbf{u}))$
min residual and features reprojected
$(\tilde{z}_\mathrm{min}(\mathbf{u}), \tilde{\mathrm{I}}_\mathrm{min}(\mathbf{u}))$
residual refinement
head scoring network
non-residual inpainting
refined center depth
$\hat{d}(\mathbf{u})$
confidence classification
center depth confidence

Figure 1: 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 1 of the main paper, 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.

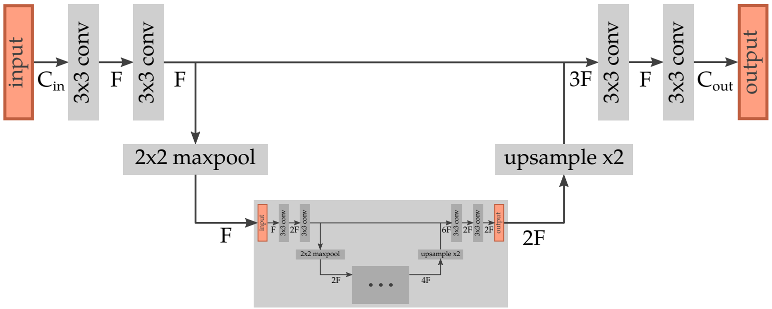

Figure 2: Convolution blocks comprise a 3×3 convolution with corresponding reflection padding, a batch normalization layer (if applicable) and a ReLU non-linearity. The number of internal channels is called F, and doubles at every level. Two convolutional groups are used instead of a UNet in the lowest resolution, while the last convolution on the highest resolution has neither a normalization nor an activation.

1.1. Building block

The building block for our network is a UNet module with skip connections, as visualized in Figure 2. At each level, the input data is processed with two convolution layers, and then downsampled with max-pooling to be processed on a lower resolution. The output from the lower resolution is bilinearly upsampled and concatenated to the processed higher-resolution features and processed by two more convolutions. On the lowest level, the down-sampled features are processed with two convolutional blocks. We write an instantiation of this building block as $\text{UNet}(C_{in}, F, C_{out}, \text{depth})$.

1.2. Neighbour reprojection

First, we briefly recap the mathematical formulations for the reprojections as given in the main text. We consider a set of N images $\mathrm{I}_n(\mathbf{u})$ with corresponding camera matrices $\mathrm{P}_n = \mathbf{K}_n\left[\mathrm{R}_n|\mathbf{t}_n\right]$, and estimated depth maps $d_n(\mathbf{u})$, where $\mathbf{u} = [u, v, 1]^\mathrm{T} \in \Omega$ is a pixel location. The 3D point corresponding with a given pixel's estimate is then given by

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

From either the bootstrap confidence network or the previous iteration's network, each depth estimate $d_n(\mathbf{u})$ has a predicted confidence. Those $\mathbf{x}_n(\mathbf{u})$ for which the input confidence is larger than 0.5 are projected onto the center view 0. We call $\mathbf{u}_{n\to m}(\mathbf{u}) = \mathrm{P}_m\mathbf{x}_n(\mathbf{u})$ the projection of $\mathbf{x}_n(\mathbf{u})$ onto neighbour m, and $z_{n\to m}(\mathbf{u}) = [0\;0\;1]\,\mathbf{u}_{n\to m}(\mathbf{u})$ its depth. The z-buffer in view 0 based on neighbour n's estimate is then

$$z_n(\mathbf{u}) = \begin{cases} \min\limits_{\mathbf{u}_n \in \Omega_n(\mathbf{u})} z_{n\to 0}(\mathbf{u}_n), & \text{if } \Omega_n(\mathbf{u}) \neq \emptyset \\ 0, & \text{elsewhere} \end{cases} \tag{2}$$

where $\Omega_n(\mathbf{u}) = \{\mathbf{u} \in \Omega | \mathrm{P}_0\mathbf{x}_n(\mathbf{u}_n) \sim \mathbf{u}\}$ is the set of pixels in view n that reproject onto $\mathbf{u}$ in view 0. Finally, we call $\mathbf{u}_n(\mathbf{u})$ the pixel in view n for which $z_n(\mathbf{u}) = z_{n\to m}(\mathbf{u}_n(\mathbf{u}))$, i.e. the pixel responsible for the entry in the z-buffer.

At the same time, we calculate the lower bound $g_\mathrm{n}(\mathbf{u})$ on the depth estimates in the center view as implied by neighbour n as the minimum depths for which the corresponding points $\mathbf{x}_0(\mathbf{u})$ are observed as empty space by neighbour n:

$$g_\mathrm{n}(\mathbf{u}) = \min\left\{d > 0 \mid d_m(\mathbf{u}_{0\to n}(\mathbf{u})) > z_{0\to n}(\mathbf{u})\right\}. \tag{3}$$

We now construct the reprojected image as the reprojected image features for the closest points behind every pixel, assuming they conform to the calculated lower bound:

$$\begin{aligned} \tilde{z}_n(\mathbf{u}) &= \begin{cases} z_n(\mathbf{u}), & \text{if } z_n(\mathbf{u}) \leq g_\mathrm{n}(\mathbf{u}) \\ 0, & \text{elsewhere} \end{cases} \\ \tilde{\mathrm{I}}_\mathrm{n}(\mathbf{u}) &= \begin{cases} \mathrm{I}_n(\mathbf{u}_n(\mathbf{u})), & \text{if } \tilde{z}_n(\mathbf{u}) > 0 \\ 0, & \text{elsewhere} \end{cases} \end{aligned} \tag{4}$$

Note that, here, counterintuively, $z_n(\mathbf{u})$ has to be smaller than or equal to the lower bound to be accepted. If it were larger then it would be either the result of bleedthrough (and should be rejected) or it would be the backside of a surface (and should also be rejected).

1.3. Neighbour pooling

The neighbour pooling modules take as input the reprojected image features $\{\mathrm{I}_n(\mathbf{u}_n(\mathbf{u}))\}$ and their corresponding depth maps $\{\tilde{z}_n(\mathbf{u})\}$, and return the mean, maximum, and minimum residual pooling (where $I(\cdot)$ is the indicator function):

$$\begin{aligned} \tilde{z}_{\text{mean}}(\mathbf{u}) &= \frac{\sum\limits_n \tilde{z}_n(\mathbf{u})}{\sum\limits_n I(\tilde{z}_n(\mathbf{u}) > 0)} \\ \tilde{\mathrm{I}}_{\text{mean}}(\mathbf{u}) &= \frac{\sum\limits_n \mathrm{I}_n(\mathbf{u}_n(\mathbf{u}))}{\sum\limits_n I(\tilde{z}_n(\mathbf{u}) > 0)} \end{aligned} \tag{5}$$

$$\begin{aligned} n^\star_{\text{max}}(\mathbf{u}) &= \underset{n}{\text{argmax}}\ \tilde{z}_n(\mathbf{u}) \\ \tilde{z}_{\text{max}}(\mathbf{u}) &= \tilde{z}_{n^\star_{\text{max}}(\mathbf{u})}(\mathbf{u}) \\ \tilde{\mathrm{I}}_{\text{max}}(\mathbf{u}) &= \mathrm{I}_{n^\star_{\text{max}}(\mathbf{u})}(\mathbf{u}_n(\mathbf{u})) \end{aligned} \tag{6}$$

$$\begin{aligned} n^\star_{\text{min}}(\mathbf{u}) &= \underset{n}{\text{argmin}}\ \|\tilde{z}_n(\mathbf{u}) - d_0(\mathbf{u})\|_1 \\ \tilde{z}_{\text{min}}(\mathbf{u}) &= \tilde{z}_{n^\star_{\text{min}}(\mathbf{u})}(\mathbf{u}) \\ \tilde{\mathrm{I}}_{\text{min}}(\mathbf{u}) &= \mathrm{I}_{n^\star_{\text{min}}(\mathbf{u})}(\mathbf{u}_n(\mathbf{u})) \end{aligned} \tag{7}$$

1.4. Shared features

Prior to the other network heads, we first calculate a set of features that will be shared by all further streams. The input to this shared feature module is $\mathrm{F}_{\text{input}}(\mathbf{u}) = \Big(\mathrm{I}_0(\mathbf{u}), d_0(\mathbf{u}), \tilde{\mathrm{I}}_{\text{mean}}(\mathbf{u}), \tilde{z}_{\text{mean}}(\mathbf{u}), \tilde{\mathrm{I}}_{\text{max}}(\mathbf{u}), \tilde{z}_{\text{max}}(\mathbf{u}), \tilde{\mathrm{I}}_{\text{min}}(\mathbf{u}), \tilde{z}_{\text{min}}(\mathbf{u})\Big)$, for a total of 16 channels. We write the structure of this module as $\text{UNet}(16, 8, 16, \text{depth} = 3)$, and call its output $\mathrm{F}_{\text{shared}}(\mathbf{u})$.

1.5. Residual refinement module

The residual refinement module acts on the original eight components plus the shared features: $(\mathrm{F}_{\text{input}}(\mathbf{u}), \mathrm{F}_{\text{shared}}(\mathbf{u}))$. The spatial support of this network is limited to have it focus on the neighbour information: its structure is given by $\text{UNet}(32, 16, 1, \text{depth} = 1)$, and its output is added to $d_0(\mathbf{u})$ to yield $\hat{d}_{\text{refined}}(\mathbf{u})$.

1.6. Inpainting module

To be able to fill in large areas with missing information, such as homogeneous areas or badly constrained areas, the inpainting module has a much larger spatial support. Similar to the residual refinement, the input is given by the 32 channels of $(\mathrm{F}_{\text{input}}(\mathbf{u}), \mathrm{F}_{\text{shared}}(\mathbf{u}))$, but now the structure is given by $\text{UNet}(32, 16, 1, \text{depth} = 5)$. Its output is called $\hat{d}_{\text{inpainted}}(\mathbf{u})$.

1.7. Head scoring network

The head scoring network weights the residual refinement and inpainting results against one another. Its inputs are hence given as $\Big(\mathrm{F}_{\text{input}}(\mathbf{u}), \mathrm{F}_{\text{shared}}(\mathbf{u}), \hat{d}_{\text{refined}}(\mathbf{u}), \hat{d}_{\text{inpainted}}(\mathbf{u})\Big)$. After the main $\text{UNet}(34, 16, 2, \text{depth} = 3)$ block, we apply a spatial Softmax layer, and use its output to weight both alternatives. The final estimate is called $\hat{d}(\mathbf{u})$, the output of our architecture.

1.8. Confidence classification

Finally, we also have the network return a prediction of its confidence. This module is intended to predict whether or not $\hat{d}(\mathbf{u})$ is within a certain threshold from the ground truth depth estimate. To do so, a $\text{UNet}(33, 16, 1, \text{depth} = 3)$ acts on $\Big(\mathrm{F}_{\text{input}}(\mathbf{u}), \mathrm{F}_{\text{shared}}(\mathbf{u}), \hat{d}(\mathbf{u})\Big)$. Its result is subsequently activated by a Sigmoid layer to yield a predicted likelihood.

1.9. Runtime

For a typical DTU scene, the timings are: 128s and 193s for the COLMAP and MVS-Net front-ends respectively, 13s per iteration for our method, 128s for COLMAP geometric consistency+fusion, and 18s for MVSNet fusion (on an NVidia Titan Xp). Therefore, our computational overhead is limited.

2. Additional results

We show more qualitative results for all elements in the test set of the DTU dataset [1] in Figure 3 and Figure 4, for COLMAP and MVSNet inputs respectively. Table 1 contains more quantative results: both the total and per-view accuracy/completeness percentages, as well as the average accuracy and completeness values for those points that are considered *relevant* (i.e. within a certain distance of the groundtruth). Table 2 contains results for the newly introduced synthetic datasets. From Table 2 it becomes evident that the synthetic datasets are easier for COLMAP (its assumptions are now fulfilled, as the evaluation does not have any sensor noise) than for MVSNet (which was trained on more realistic images). Applying our proposed fusion steps significantly improves the results in nearly all cases. These evaluations run on quarter-scale input (480×270 rather than 1920×1080), but we use the original, high-resolution, point clouds as a reference for these comparison.

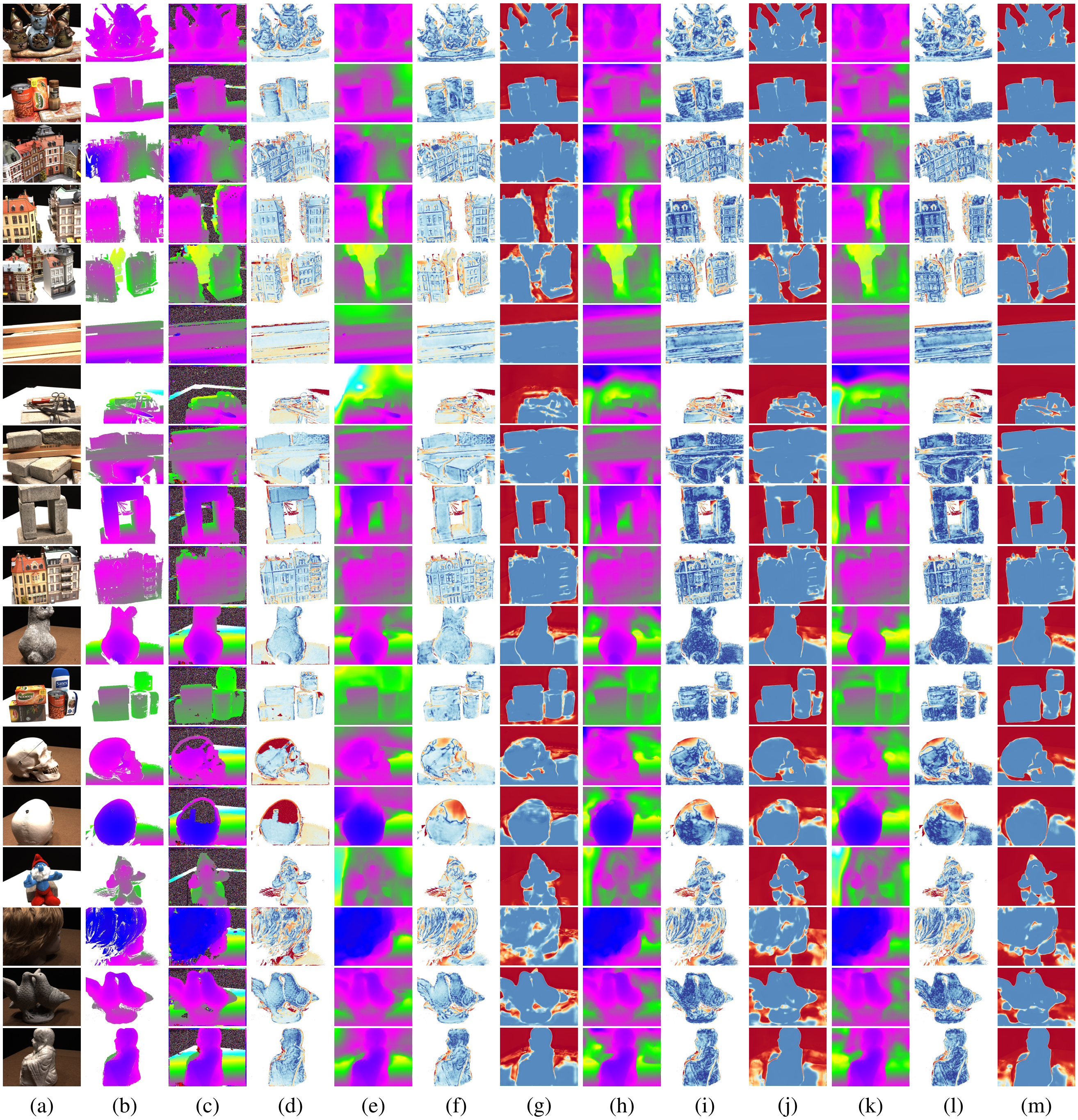

Figure 3: Qualitative results for our approach on the COLMAP inputs, for all elements in the test set. For a given input image (a) we show first the ground truth image (b), the input estimate (c) and its error (d). The subsequent columns show the output estimate, error and trust for three iterations. The error visualization works logarithmically: at error value 5, the colour is white. At infinite error, a pixel is shown as dark red and at zero error it is displayed a dark blue. While the network returns nonsense values for the badly constrained areas (there is no information nor supervision available), these areas are correctly filtered out with the trust estimate. Best viewed digitally.

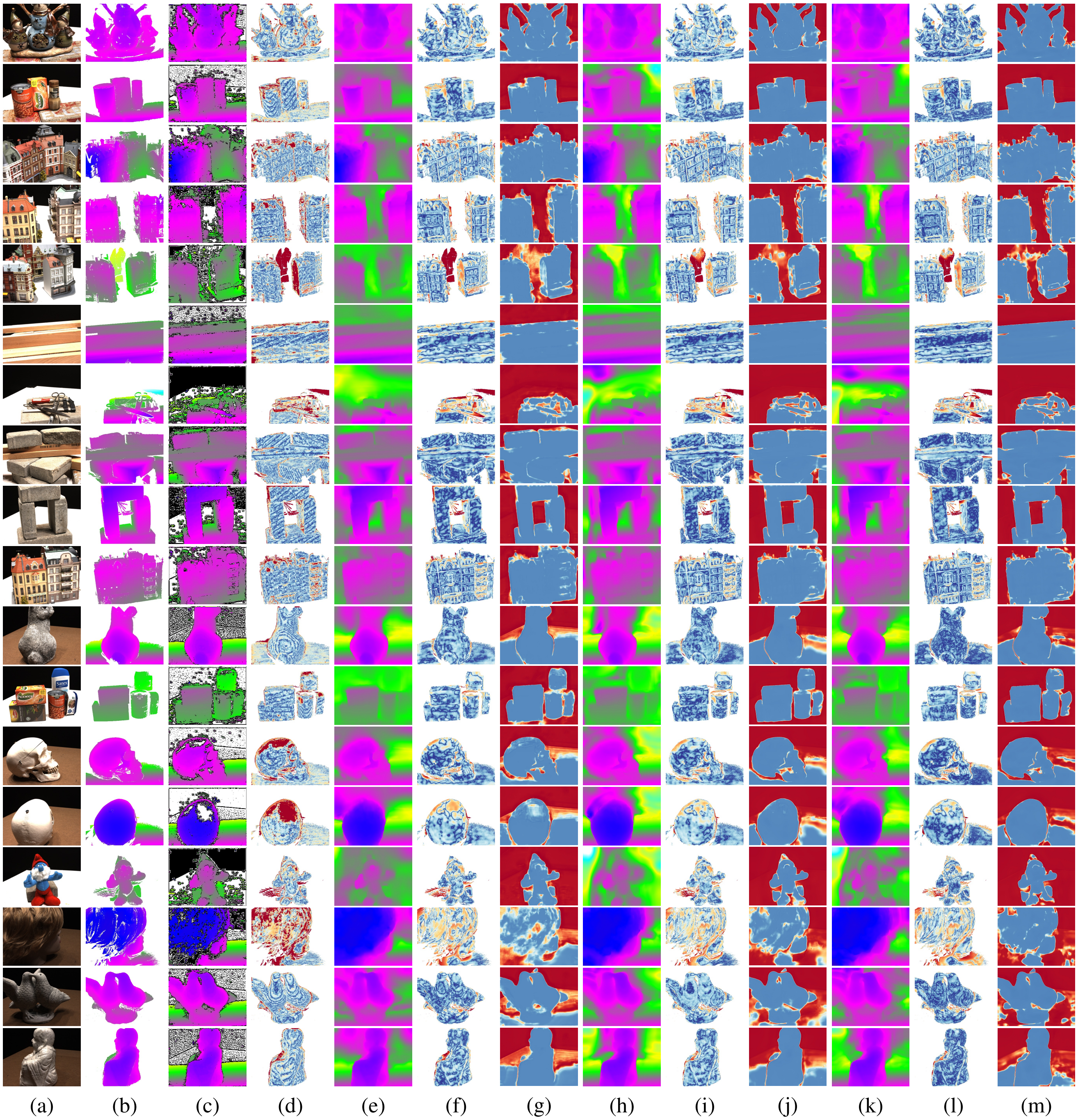

Figure 4: Qualitative results for our approach on the MVSNet inputs, for all elements in the test set. For a given input image (a) we show first the ground truth image (b), the input estimate (c) and its error (d). The subsequent columns show the output estimate, error and trust for three iterations. The error visualization works logarithmically: at error value 5, the colour is white. At infinite error, a pixel is shown as dark red and at zero error it is displayed a dark blue. While the network returns nonsense values for the badly constrained areas (there is no information nor supervision available), these areas are correctly filtered out with the trust estimate. Best viewed digitally.

Table 1: Quantitative evaluation for the COLMAP [2] and MVSNet [3] front-ends followed by COLMAP fusion. Using 12 neighbours, selected using a mixed near-far strategy, and three refinement iterations. We report the numbers in the paper both for $\tau_d = 1$ and $\tau_d = 2$ (all values reported here are expected to be lower for lower τ_d). For the distances in mm, lower is better; for the percentages, higher is better.

	per view						full cloud					
Threshold $\tau_d = 1.0$	accuracy (mm)	completeness (mm)	chamfer (mm)	accuracy (%)	completeness (%)	mean (%)	accuracy (mm)	completeness (mm)	chamfer (mm)	accuracy (%)	completeness (%)	mean (%)
COLMAP [2]	0.51	0.68	0.59	29.5	24.4	26.9	**0.40**	0.69	0.55	59.4	36.7	48.1
COLMAP [2] + ours	**0.41**	**0.63**	**0.52**	**61.8**	**49.1**	**55.4**	0.42	**0.38**	**0.40**	**69.4**	**65.5**	**67.9**
MVSNet [3]	0.45	0.66	0.55	29.5	23.5	26.5	**0.39**	0.41	0.40	**76.7**	**74.0**	**75.4**
MVSNet [3] + ours	**0.39**	**0.62**	**0.50**	**66.4**	**24.8**	**45.6**	0.40	**0.37**	**0.38**	74.4	65.5	70.0

	per view						full cloud					
Threshold $\tau_d = 2.0$	accuracy (mm)	completeness (mm)	chamfer (mm)	accuracy (%)	completeness (%)	mean (%)	accuracy (mm)	completeness (mm)	chamfer (mm)	accuracy (%)	completeness (%)	mean (%)
COLMAP [2]	0.96	1.17	1.06	62.5	51.5	56.0	**0.59**	1.03	0.81	72.6	72.0	72.3
COLMAP [2] + ours	**0.60**	**0.94**	**0.77**	**83.9**	**65.9**	**75.9**	0.71	**0.57**	**0.64**	**81.2**	**72.4**	**76.8**
MVSNet [3]	0.74	1.06	0.90	76.0	**34.8**	55.9	**0.57**	**0.55**	**0.56**	**88.3**	**66.6**	**77.4**
MVSNet [3] + ours	**0.55**	**0.91**	**0.73**	**92.2**	34.0	**63.6**	0.65	0.56	0.60	86.3	65.5	75.9

Table 2: Quantitative evaluation on the two synthetic datasets. For UnrealDTU, we pre-train on FlyingThingsMVS. The values for tau_d were chosen so as to make the percentages discriminative.

			per view			full cloud		
			accuracy (%)	completeness (%)	mean (%)	accuracy (%)	completeness (%)	mean (%)
FT MVS	$\tau_d = 0.02$	COLMAP [2]	87	65	76	77	78	78
		COLMAP [2] + ours	**91**	**73**	**82**	**86**	**94**	**90**
		MVSNet [3]	55	50	52	**79**	62	70
		MVSNet [3] + ours	**77**	**61**	**69**	71	**83**	**77**
UnrealDTU	$\tau_d = 0.4$	COLMAP [2]	83	75	79	69	60	64
		COLMAP [2] + ours	**93**	**80**	**86**	**86**	**97**	**92**
		MVSNet [3]	38	12	25	53	77	65
		MVSNet [3] + ours	**82**	**68**	**75**	**70**	**95**	**82**

References

[1] R. R. Jensen, A. L. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. Large scale multi-view stereopsis evaluation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 3

[2] 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. 6

[3] 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. 6