Supplementary Material for DeepVideoMVS: Multi-View Stereo on Video with Recurrent Spatio-Temporal Fusion

A. Training Details

We use an image size of 256×256 with cropping and scaling. Since the original image resolution is 640×480 in the ScanNet dataset, we crop the image from the left and right equally to acquire an image size of 480×480, then downscale to 256×256. During training, we compute the cost volume for a reference frame using one measurement frame.

Pair Network Training. We use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a constant learning rate of $1\mathrm{e}{-4}$. The pair network is trained with a mini-batch size of 14 for 600K iterations in total. We load the pretrained weights for the MnasNet layers, which are supplied by PyTorch, and freeze these layers for the first 200K iterations. We predict depth maps for both of the input images at each forward pass, *i.e.*, after the shared feature extraction modules, we consider a given feature map once as the reference feature map and once as the measurement feature map.

Fusion Network Training. The fusion network is trained with a subsequence length of 8 and a mini-batch size of 4. As the initial weights for the feature extraction layers, the feature pyramid network (FPN) and the encoder, we use the checkpoint saved at 100K iterations of the pair network training. We freeze the parameters for these modules and train only the ConvLSTM cell and the decoder for 25K iterations, which are randomly initialized. Next, we add the FPN and the encoder weights among the trainable parameters and train for another 25K iterations. Then, we unfreeze the MnasNet layers and train the whole network up to 1000K iterations, while validating and saving checkpoints frequently for early stopping. During these iterations, we use the groundtruth depth map to warp the hidden state. Finally, we load the best checkpoint and finetune only the cell for another 25K iterations with a learning rate of $5\mathrm{e}{-5}$ while warping the hidden states with the predicted depth maps. We do not allow the gradients to flow through the depth map prediction that warps the hidden state. As our attempts suggest, allowing this gradient flow introduces a complex causal relationship among the predictions and the gradients explode when the training experiences a large loss in an arbitrary subsequence.

Data Augmentation. We use several data augmentation techniques. These are applying slight changes to the color space of the images, randomly reversing the order of the frames in a subsequence, and randomly changing the geometric scale of the subsequences. Also, we apply random horizontal flips to the cost volume and the extracted features during pair network training to increase the diversity of the cases that the encoder and the decoder experience. For the geometric scale augmentation, we choose the default sampling interval of the scaling factor as $[0.666, 1.5]$. When the training pipeline requests a new subsequence from the data pipeline, we adjust these bounds according to the minimum and the maximum depth measurements in the subsequence. This ensures that even after applying a random geometric scaling, the cost volume can still contain a correct local extremum for every pixel. The sampled scale factor is multiplied with the groundtruth depth values and the translation columns of the camera poses in an entire subsequence.

B. Evaluation Metrics

For a predicted depth map, the metrics are calculated as:

i. abs = $\frac{1}{N}\sum_{i=1}^{N}|d_i - \hat{d}_i|$

ii. abs-rel = $\frac{1}{N}\sum_{i=1}^{N}\frac{|d_i - \hat{d}_i|}{d_i}$

iii. abs-inv = $\frac{1}{N}\sum_{i=1}^{N}\left|\frac{1}{d_i} - \frac{1}{\hat{d}_i}\right|$

iv. inlier ratio = $\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\left[\frac{d_i}{1.25} < \hat{d}_i < 1.25 \times d_i\right]$

where $\hat{d}_i$ and d_i denote the predicted depth value and the groundtruth depth value for a given pixel i that has a valid depth measurement, N is the number of pixels that have valid depth measurements, and $\mathbb{1}$ is the indicator function.

C. Inference Time and Memory Consumption

The average duration of a single forward pass and the GPU memory consumption for the methods in comparison are given in Tab. 5. The measurements are taken on a workstation with NVIDIA GTX 1080Ti GPU. We do not consider the time spent by the CPU or the transfer time of the data to/from the GPU. We fix the input image sizes for all methods to 320×256, start recording the times after the first 100 forward passes (warm start), and always start a method when the CPU and GPU temperature are below 40°C. We run all the methods with a mini-batch size of 1, *i.e.*, the simulation of predicting a single depth map for the current time step. For DELTAS, we replace the PyTorch's singular value decomposition with a custom function that the authors

suggest to increase the inference speed. However, note that we could not reproduce the inference times that the authors of DELTAS report as around 90 milliseconds on NVIDIA Titan RTX GPU.

	Time(ms) ↓	FPS ↑	Memory(MB) ↓
MVDepth	45.84	∼21.8	1081
GPMVS	47.81	∼20.9	1083
DPSNet	172.62	∼5.8	863
NRGBD	192.43	∼5.2	1485
*DELTAS	374.78	∼2.7	2371
Ours (Pair)	29.72	∼33.7	795
Ours (Fusion)	37.14	∼26.9	1061

Table 5: Mean inference time and GPU memory consumption of all methods. Inference time is averaged over 300 forward passes. All methods are run with an image size of 320×256 and use one measurement frame, except Neural RGBD that requires minimum two measurement frames. * the authors report faster times.

D. Evaluation Set Details

We use four real and one synthetic dataset to evaluate the methods. All of the datasets provide RGB-D videos at 640×480 depth image resolution and 3D camera poses. The selected sequences are as follows:

- **ScanNet** [11]: All 100 official test sequences, *i.e.*, scene0707 to scene0806.
- **7-Scenes** [16]: chess-01, chess-02, fire-01, fire-02, head-02, office-01, office-03, pumpkin-03, pumpkin-06, redkitchen-01, redkitchen-02, stairs-02, stairs-06.
- **RGB-D Scenes V2** [28]: scene-01, scene-02, scene-05, scene-06, scene-09, scene-10, scene-13, and scene-14.
- **TUM RGB-D SLAM** [47]: fr1-desk, fr1-plant, fr1-room, fr1-teddy, fr2-desk, fr2-dishes, fr2-large-no-loop, fr3-cabinet, fr3-long-office-household, fr3-nostructure-notexture-far, fr3-nostructure-texture-far, fr3-structure-notexture-far, fr3-structure-texture-far.
- **Augmented ICL-NUIM** [18, 48]: livingroom1, livingroom2, office1, office2.

E. Additional Ablation Studies

In this section, we provide additional ablation studies. We present the results acquired on our *validation split* of the ScanNet dataset. These experiments were conducted at different stages of this work. Each subsection is self-contained, *i.e.* the models in a given subsection are trained with identical training and data pipelines. However, the results cannot be meaningfully compared in between subsections, since they may employ different training strategies.

Configuration-1
ELU activation, no normalization

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{w}_{xi} * \mathbf{X}_t + \mathbf{w}_{hi} * \mathbf{H}_{t-1}) \\
\mathbf{f}_t &= \sigma(\mathbf{w}_{xf} * \mathbf{X}_t + \mathbf{w}_{hf} * \mathbf{H}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{w}_{xo} * \mathbf{X}_t + \mathbf{w}_{ho} * \mathbf{H}_{t-1}) \\
\mathbf{g}_t &= ELU(\mathbf{W}_{xg} * \mathbf{X}_t + \mathbf{w}_{hg} * \mathbf{H}_{t-1}) \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{H}_t &= \mathbf{o}_t \odot ELU(\mathbf{C}_t)
\end{aligned}$$

Configuration-2
tanh activation, no normalization

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{w}_{xi} * \mathbf{X}_t + \mathbf{w}_{hi} * \mathbf{H}_{t-1}) \\
\mathbf{f}_t &= \sigma(\mathbf{w}_{xf} * \mathbf{X}_t + \mathbf{w}_{hf} * \mathbf{H}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{w}_{xo} * \mathbf{X}_t + \mathbf{w}_{ho} * \mathbf{H}_{t-1}) \\
\mathbf{g}_t &= tanh(\mathbf{w}_{xg} * \mathbf{X}_t + \mathbf{w}_{hg} * \mathbf{H}_{t-1}) \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{H}_t &= \mathbf{o}_t \odot tanh(\mathbf{C}_t)
\end{aligned}$$

Configuration-3
scaled tanh activation, no normalization

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{w}_{xi} * \mathbf{X}_t + \mathbf{w}_{hi} * \mathbf{H}_{t-1}) \\
\mathbf{f}_t &= \sigma(\mathbf{w}_{xf} * \mathbf{X}_t + \mathbf{w}_{hf} * \mathbf{H}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{w}_{xo} * \mathbf{X}_t + \mathbf{w}_{ho} * \mathbf{H}_{t-1}) \\
\mathbf{g}_t &= \alpha \times tanh(\mathbf{w}_{xg} * \mathbf{X}_t + \mathbf{w}_{hg} * \mathbf{H}_{t-1}) \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{H}_t &= \mathbf{o}_t \odot (\alpha \times tanh(\mathbf{C}_t))
\end{aligned}$$

Configuration-4
ELU activation, layer normalization before all act.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(layernorm(\mathbf{w}_{xi} * \mathbf{X}_t + \mathbf{w}_{hi} * \mathbf{H}_{t-1})) \\
\mathbf{f}_t &= \sigma(layernorm(\mathbf{w}_{xf} * \mathbf{X}_t + \mathbf{w}_{hf} * \mathbf{H}_{t-1})) \\
\mathbf{o}_t &= \sigma(layernorm(\mathbf{w}_{xo} * \mathbf{X}_t + \mathbf{w}_{ho} * \mathbf{H}_{t-1})) \\
\mathbf{g}_t &= ELU(layernorm(\mathbf{w}_{xg} * \mathbf{X}_t + \mathbf{w}_{hg} * \mathbf{H}_{t-1})) \\
\mathbf{C}_t &= layernorm(\mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t) \\
\mathbf{H}_t &= \mathbf{o}_t \odot ELU(\mathbf{C}_t)
\end{aligned}$$

Configuration-5
ELU activation, layer normalization before only ELU act.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{w}_{xi} * \mathbf{X}_t + \mathbf{w}_{hi} * \mathbf{H}_{t-1}) \\
\mathbf{f}_t &= \sigma(\mathbf{w}_{xf} * \mathbf{X}_t + \mathbf{w}_{hf} * \mathbf{H}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{w}_{xo} * \mathbf{X}_t + \mathbf{w}_{ho} * \mathbf{H}_{t-1}) \\
\mathbf{g}_t &= ELU(layernorm(\mathbf{w}_{xg} * \mathbf{X}_t + \mathbf{w}_{hg} * \mathbf{H}_{t-1})) \\
\mathbf{C}_t &= layernorm(\mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t) \\
\mathbf{H}_t &= \mathbf{o}_t \odot ELU(\mathbf{C}_t)
\end{aligned}$$

Figure 8: Various activation and normalization options for the ConvLSTM cell. Configuration-5 delivers the best depth prediction performance, while maintaining a stable behaviour.

Activation and Normalization in the ConvLSTM Cell. There are many activation and normalization options that can be included in a ConvLSTM cell. The options that showed significant effects are provided in Fig. 8 and the corresponding depth prediction performances are given in Tab. 6.

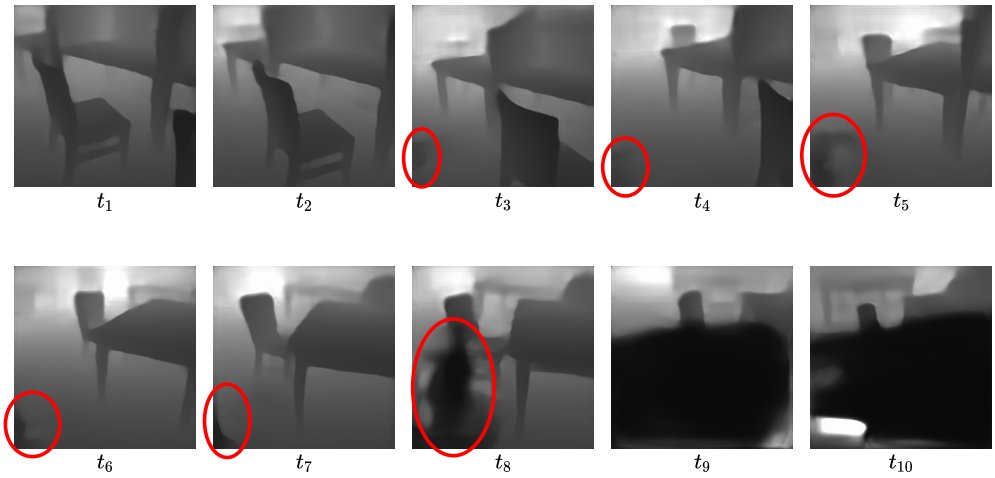

Figure 7: Placing ELU activations without any normalization in the ConvLSTM cell causes stability issues. The noise starting at the lower left corner of the hidden state (thus the prediction), gets propagated through time and grows with the subsequent convolution operations.

	abs	abs-rel	abs-inv	noise test	loop test
Configuration-1	0.1267	0.0580	0.0344	✗	✗
Configuration-2	0.1321	0.0601	0.0351	✓	✓
Configuration-3	0.1272	0.0590	0.0351	✓	✓
Configuration-4	0.1236	0.0578	0.0347	✓	✓
Configuration-5	**0.1206**	**0.0567**	**0.0339**	✓	✓

Table 6: Evaluation of different activation and normalization options inside the ConvLSTM cell.

Placing ELU activations enables the range of values to be similar at the output of the encoder and the output of the ConvLSTM cell. Whereas, tanh limits the range of the hidden state to $[-1, 1]$. During training, we observe that ELU (Configuration-1) achieves a better learning curve and provides better validation scores than tanh (Configuration-2). However, since we validate on short subsequences, similar to the training subsequences, we can not observe the side effects of the ELU activation during training. When the validation is run on long sequences, we observe that naively placing ELU activations causes two issues. Both of them are due to the output range $[-1, \infty]$ of ELU activation, which is asymmetric and unbounded on the positive side.

The first issue is that, if the model fails to produce a "good" hidden state at an arbitrary time, and instead computes a slightly noisy state, the noise gets propagated and grows incrementally, causing whole predictions to diverge over time. This issue is depicted in Figure 7. It can occur at any arbitrary time in a sequence, both with and without the warping of the hidden state. Randomly introducing positive additive noise to the hidden state consistently reproduced this issue and served as a test of robustness for all the activation and normalization configurations. We call this the "noise test". The second issue is observed while simulating online multi-view stereo captures with tens of thousands of frames, done by running the inference on the sequences in our validation split that include pose loops. We call this the "loop test". With ELU activation and no normalization, the cell state ($\mathbf{C}_t$) is mostly increasing. In very long sequences, the values in the states ($\mathbf{H}_t$, $\mathbf{C}_t$) eventually reach very large values, which destabilizes the model and the predictions diverge similar to the first issue.

Scaling the tanh activation function (Configuration-3) with a learnable α, which the training ends up assigning 4.459, adjusts the numerical range of the hidden state and results in better depth predictions than unscaled tanh. However, it can not achieve the accuracy of the unstable Configuration-1. Whereas, Configuration-4 gives similar results to Configuration-1, while also passing the stability test and the loop test. In this configuration, the sparse nature of ELU activations are kept while the layer normalization ensures that the cell state always have a zero mean and unit variance per channel, and the values do not grow uncontrollably. We further improve the performance by removing the layer normalizations before the sigmoid activations as in Configuration-5. Placing layer normalizations before the sigmoid activations dictate that, in $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$ gate tensors, roughly half of the values are always below 0.5 and half of them are above 0.5. This unnecessarily constrains the learning, thus removing them results in better depth predictions.

ConvLSTM vs. ConvGRU. We compare ConvGRU and ConvLSTM cells as the recurrent cell choice for our proposed fusion module. ConvLSTM cell logic is given in Eq. 5 in the paper. By applying the principles learned from the ablation studies on activation and normalization options, ConvGRU cell is set up as

$$\begin{aligned}
\mathbf{u}_t &= \sigma(\mathbf{w}_{xu} * \mathbf{X}_t + \mathbf{w}_{hu} * \mathbf{H}_{t-1}) \\
\mathbf{r}_t &= \sigma(\mathbf{w}_{xr} * \mathbf{X}_t + \mathbf{w}_{hr} * \mathbf{H}_{t-1}) \\
\mathbf{o}_t &= \text{ELU}(\text{layernorm}(\mathbf{w}_{xo} * \mathbf{X}_t + \mathbf{w}_{ho} * (\mathbf{H}_{t-1} \odot \mathbf{r}_t))) \\
\mathbf{H}_t &= \text{layernorm}(\mathbf{u}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{u}_t) \odot \mathbf{o}_t). \qquad (10)
\end{aligned}$$

We test the performances of these cells while warping the hidden states during propagation. Tab. 7 presents the study results where ConvLSTM outperforms its counterpart by 5.6% in abs-inv error. We speculate that warping the *sole* hidden state in ConvGRU hinders the memory functionality. Since each warping operation removes information from non-overlapping view frustums of consecutive frames externally, such info gets lost irrecoverably in the ConvGRU case. Whereas, with ConvLSTM, we take advantage of the two internal states and manipulate only the hidden state.

	abs	abs-rel	abs-inv
Fusion with ConvLSTM (Eq. 5) and Warping	**0.1192**	**0.0565**	**0.0340**
Fusion with ConvGRU (Eq. 10) and Warping	0.1248	0.0594	0.0359

Table 7: Comparison of ConvLSTM and ConvGRU cell performances when placed in the proposed fusion module.

Finetuning the Cell. In Tab. 8, B already provides a high performance, which shows that the model does not require pixel-perfect warpings while propagating the hidden states. Nevertheless, finetuning the ConvLSTM cell with the test time strategy reduces the discrepancy between using the groundtruth depth and predicted depth, *c.f*. relative differences between A, B and C, D. Surprisingly, we also observe a slight improvement from A to C.

	Warp With	abs	abs-rel	abs-inv
Before Finetuning (**A**)	Groundtruth	0.1192	0.0565	0.0340
Before Finetuning (**B**)	Prediction	0.1207	0.0573	0.0346
After Finetuning (**C**)	Groundtruth	0.1189	0.0563	0.0337
After Finetuning (**D**)	Prediction	0.1191	0.0564	0.0338

Table 8: Effect of finetuning the cell while warping the hidden states with predictions instead of groundtruths.

Skip Connections from Feature Pyramid to Encoder. As presented, we pass multi-scale image features to the cost volume encoder by placing several skip connections from the feature pyramid network. In order to measure the effect of these connections, we remove the low resolution skip connections and keep only the half resolution feature map coming from the feature pyramid, *i.e*., single-skip design. As shown in Tab. 9, despite achieving similar results in a pair network, multi-skip design is around 4.5% better in the abs-inv metric when placed in a fusion network. We conjecture that we mostly benefit from a secondary effect rather than the primary motivation of these connections which is to guide the encoder with strong cues about the image content at each level. The secondary effect is the established phenomenon that skip connections ease the training and smooth the loss landscape [30]. Thus, the multi-skip design help the model reach a better loss minimum in the presence of a ConvLSTM (which typically complicates the loss landscape) by providing more "gradient highways".

	abs	abs-rel	abs-inv
Pair Network - Multi-Skip Design	0.1441	0.0695	0.0427
Pair Network - Single-Skip Design	0.1444	0.0692	0.0426
Fusion Network - Multi-Skip Design	0.1192	0.0565	0.0340
Fusion Network - Single-Skip Design	0.1245	0.0589	0.0355

Table 9: Effect of placing multiple skip connections from the feature pyramid to the cost volume encoder.

Accuracy vs. Time Trade-off of the Fusion. We show the effectiveness of our proposed fusion in an alternative way in Tab. 10, where we assume a fixed budget for the inference time. We re-train the pair network with the number of sweeping planes M=88 to increase the sweep resolution while compensating the additional runtime of the fusion. The increase in the depth resolution improves the pair network's performance as expected, but we only see 1.2% improvement in abs-inv on our ScanNet validation split. The proposed fusion, with a similar overhead of computational time, results in 20.4% improvement. This demonstrates the high skew in the accuracy vs. time trade-off.

	abs	abs-rel	abs-inv	time (ms)
Pair Network with M=64	0.1441	0.0695	0.0427	29.72
Pair Network with M=88	0.1429	0.0688	0.0422	37.56
Fusion Network with M=64	0.1192	0.0565	0.0340	37.14

Table 10: Comparison between increasing the plane sweep resolution and employing the proposed fusion mechanism.

F. Additional Qualitative Results and Supplementary Video

In Fig. 9 and Fig. 10, we provide additional example depth predictions. The same trends we discussed in the main paper are observed in these examples too. Our pair network can already output plausible depth predictions, and the proposed fusion module improves the coherency of the depth values for an image leveraging the past information. This effect is better observed in the supplementary video, https://ardaduz.github.io/deep-video-mvs/miscellaneous/deep-video-mvs-supplementary-video.mp4, in which we demonstrate the predicted depth map sequences and the resulting TSDF reconstructions of several indoor scenes. Our spatio-temporal fusion network produces predictions with less flickering effects in between time steps, it achieves a superior geometric consistency than our pair network and the existing methods. We noticeably output more consistent depth predictions for the planar surfaces throughout a sequence which gets reflected as smooth reconstructions of such surfaces.

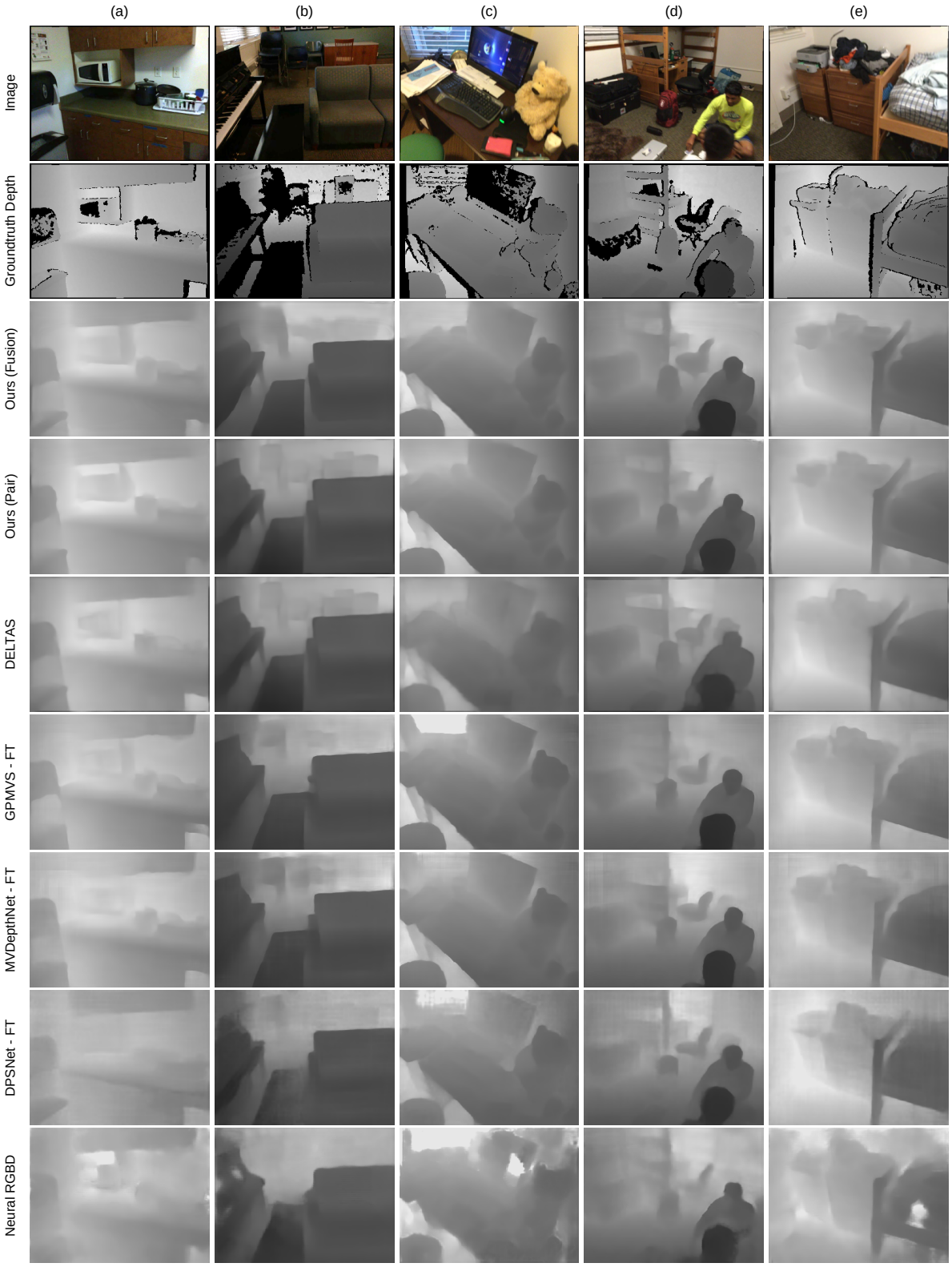

Figure 9: Example depth predictions from ScanNet.

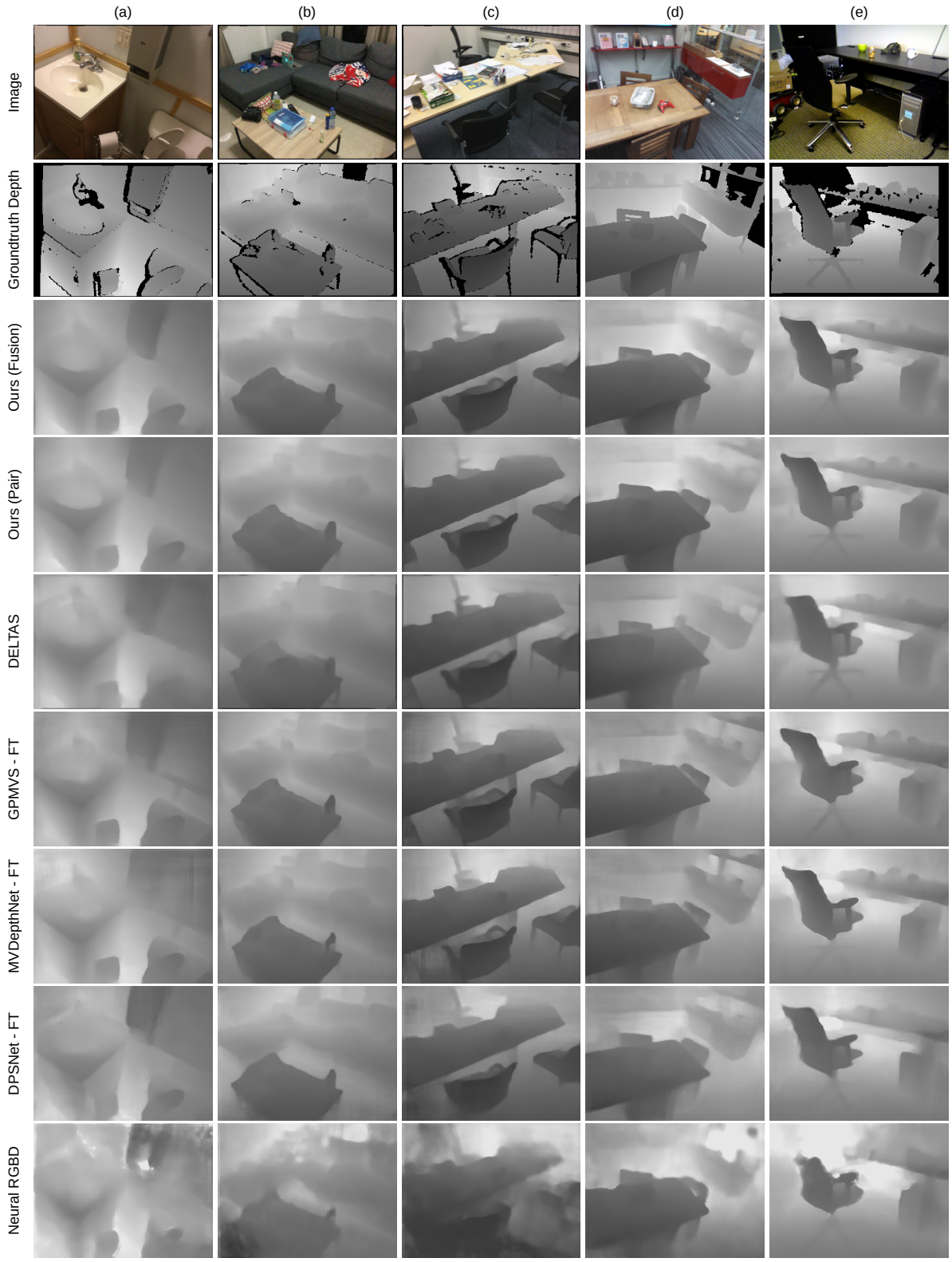

Figure 10: Example depth predictions from ScanNet, 7-Scenes and RGB-D Scenes V2.