NeuralFusion: Online Depth Fusion in Latent Space — Supplementary Material —

Silvan Weder[1] Johannes L. Schönberger[2] Marc Pollefeys[1,2] Martin R. Oswald[1]
[1]Department of Computer Science, ETH Zurich
[2]Microsoft Mixed Reality and AI Zurich Lab

Abstract

This supplementary material accompanies the main publication of – NeuralFusion: Online Depth Fusion in Latent Space – providing further information for better reproducibility as well as additional evaluations and qualitative results. We also provide a video explaining the main method as well as showcasing the most important results.

A. Evaluation Metrics

In order to compare our method to state-of-the-art learning-based methods and standard TSDF fusion, we compute the following six metrics on reconstructions:

Mean Squared Error (MSE) and Mean Absolute Distance (MAD). The mean squared error measures the reconstruction error on the TSDF field by penalizing large surface deviations and outliers. The mean absolute distance is also computed on the TSDF grid. However, it mainly quantifies the performance on reconstructing fine geometric details.

Accuracy (Acc.), F1 Score, Intersection-over-Union (IoU): The accuracy is computed over the occupancy obtained from the sign of the TSDF grid. We also report the F1 score, which is the harmonic mean of precision and recall. By measuring both, completeness and accuracy, it is a more holistic metric for quantifying the performance of a reconstruction method. Moreover, we measure the IoU on the occupancy grid. The IoU especially quantifies artifacts typically encountered in reconstructions from noisy depth maps, such as surface and corner thickening and the vanishing of fine geometric details.

Mesh Completeness (M.C.) and Accuracy (M.A.) We compute the completeness using the evaluation pipeline from [2]. The completeness describes the distance from points sampled on the ground-truth mesh to the closest point on the reconstructed mesh. Vice-versa, the accuracy computes the distance from points sampled on the reconstructed mesh to the closest point on the ground-truth mesh.

B. Reproducibility

For reproducibility, our source code will be made publicly available upon publication. We further present more details of our fusion pipeline in the following.

B.1. Details on Pipeline Architecture

Our method consists of four neural network components that are used for **(i)** sub-volume **extraction** of the global canonical feature volume, **(ii) fusion** of previously fused feature with a new depth map , **(iii) integration** of the fused updates back into the global feature volume, and **(iv) translation** from the latent feature space to TSDF and occupancy.

(i) Extraction Layer. In the extraction layer, we extract the current state of the global feature volume into a view-dependent canonical feature volume defined by the camera parameters of the current measurement. In a first step, we un-project all depth pixels into the global feature grid using the camera parameters:

$$p_{XYZ} = \begin{bmatrix} R & t \end{bmatrix}^{-1} \begin{bmatrix} K^{-1}p & 1 \end{bmatrix} \tag{1}$$

where p_{XYZ} are the coordinates of the un-projected point in world coordinates and $p = (p_x, p_y, d)^T$ are the pixel coordinates with its corresponding depth measurements. In a second step, we sample points around p_{XYZ} in a window centered at p_{XYZ} and aligned with the direction of the viewing ray. This procedure is inspired by the extraction step used in [3]. Finally, we convert the coordinates of each sampled point to grid coordinates and extract the current feature state using nearest-neighbor interpolation.

(ii) Feature Fusion Network. The feature fusion network consists of three components: Feature Encoder, Feature Decoder, and Feature Normalization, which are detailed in the following.

1. **Feature Encoder:** The feature encoder is built from four network blocks each consisting of the following modules: 1) a 2D convolution having kernel size of 3 and zero padding reducing the number of input channels, 2) a layer normalization, 3) tanh activation, 4)

again a 2D convolution having kernel size of 3 and zero padding but without reducing the channels, 5) layer normalization, and 6) tanh activation. The ouput of each block is concatenated with its input and passed to the next block.

2. **Feature Decoder:** The feature decoder also consists of four neural blocks, of which each has the same design as the neural blocks in the feature encoder. However, instead of having a kernel-size of three, the convolutional layers in the feature decoder have a kernel-size of one. The motivation behind this choice is that the encoder has already encoded enough neighboring information and, therefore, the decoder predicts the feature updates based on the encoded information for each ray separately.

3. **Feature Normalization:** After predicting the feature updates for each position in the local, view-dependent feature volume, we normalize each feature vector. This normalization prevents the feature values from becoming too large and, therefore, it improves the pipeline's capability to update the scene.

(iii) Integration Layer. In the integration layer, we integrate the predicted feature updates from the fusion network back into the global feature volume. Therefore, we aggregate all updates that are mapped to the same global feature grid location using the correspondence given by the camera parameters. Then, we use an average pooling to combine multiple correspondences to the same grid location. Finally, we update the feature volume by using a running average update similar to [1].

(iv) Feature Translation Network. The feature translation network renders the output modalities (TSDF and occupancy) from the latent feature representation for a specific query point p_i. It consists of three components: a neighborhood interpolator, a translation MLP, and two network heads predicting the output modalities.

1. **Neighborhood Interpolator:** The neighborhood interpolator encodes information from the neighboring feature vector into a single feature vector. Therefore, all neighboring feature vectors are concatenated and passed through a single linear layer followed by tanh activation. The output has the same dimension as one single feature vector.

2. **Translation MLP:** The output of the neighborhood interpolator is concatenated with the query point feature vector $g_t(p_i)$ and passed through the translation MLP. The translation MLP is built from four linear layers interleaved with tanh activations. During training, the output of each layer is further passed through a channel-wise dropout layer with dropout probability $p = 0.2$. The first layer has 32 output channels, the second layer has 16 output channels, and the third and fourth layer have each 8 output channels. The output of each layer is concatenated with the query point feature vector. The output of the final layer is then passed to the two network output heads.

3. **Network Output Heads:** The translation network has two network output heads. Each head takes the output of the MLP as an input and predicts a translation modality. The TSDF head predicts the TSDF using a single linear layer outputting one channel that is followed by a tanh activation. The output of the activation is further scaled by the truncation band of the ground-truth TSDF (0.04) to map it into the correct value range. The occupancy head is also passed through a linear layer but activated using a sigmoid activation to map into a unit interval.

B.2. Details on Hyperparameters

Table 1 summarizes the choice of all hyperparameters that we used for all experiments in our work.

Optimizer	
Name	ADAM
β_1	0.9
β_2	0.999
ϵ	$1.e-08$
Learning Rate Scheduling	
Initial Learning Rate	0.01
Decay	0.998
Loss Weights	
λ_1	1.0
λ_2	10.0
λ_o	0.01
λ_g	0.05

Table 1: Network hyperparameters

C. Qualtitative Results

In this section, we present more qualitative results to demonstrate the performance of our method.

C.1. Synthetic Data

In Figure 2, we show additional qualitative results for the outlier robustness experiment that we presented in the main paper. Our method is consistently better in filtering outliers than existing methods that have no outlier filtering or filter outliers heuristically.

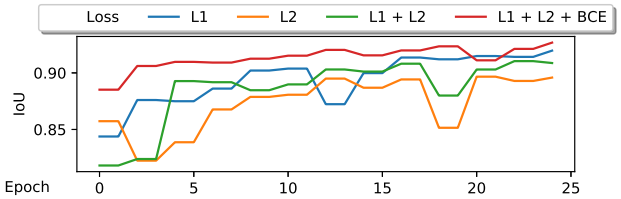

Figure 1: **Loss Ablation.** All losses combined yield best results.

C.2. Real-World Data

In Figure 3 we show more results on the real-world Scene3D [5] dataset. With this experiment, we demonstrate that our method generalizes well to real-world data and is able to fuse and reconstruct measurements of large-scale real-world scenes.

D. Further Evaluation

D.1. Generalization from a Single Object

In order to demonstrate the compactness and generalization performance of our network, we train it only on a single chair object from the ModelNet [4] dataset. We augment the input depth maps with artificial noise of scale 0.01. We report the results in Table 2 and show that our method trained on a single object achieves almost the same performance as our method trained on the full training set. Moreover, it outperforms the currently best performing method - RoutedFusion [3] - that is trained on the full training set. This result

Method	**MSE↓** [e-05]	**MAD↓** [e-02]	**Acc.↑** [%]	**IoU↑** [0,1]
RoutedFusion [3] (full training set)	6.79	0.56	94.44	0.821
Ours (full training set)	4.84	**0.42**	**96.30**	**0.874**
Ours (single object)	**3.94**	0.44	94.51	0.848

Table 2: Our method trained on the standard training split and on a single chair object only. The model trained on a single object is almost on par with our model trained on the full training set and outperforms the next best existing method trained on the full training set.

indicates the applicability of our method to many real-world scenarios, where the sensor setup might change. In fact, only very little training data is required to retrain our method and achieve state-of-the-art reconstruction results.

D.2. Loss Ablation

We have also run an ablation study to evaluate the importance of the different terms in our loss function. In Figure 1, we show that the combination of all three loss terms yields best results. The binary cross entropy is particularly useful to improve convergence in the beginning of the training as the network learns to predict a coarse shape that is further refined by the losses on the SDF as training progresses.

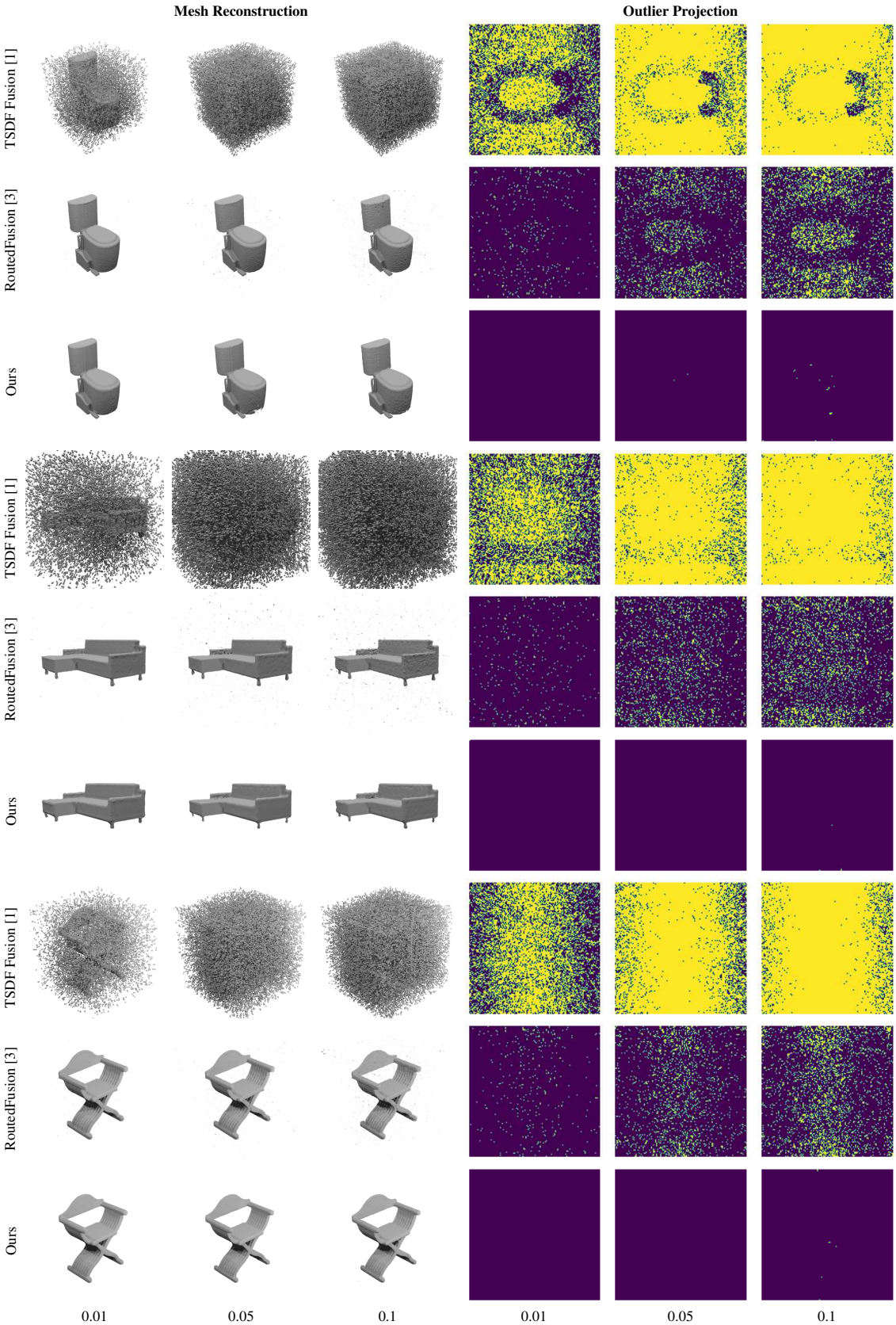

Figure 2: **More qualitative results for different outlier fractions on ModelNet [4] examples.** Our method consistently removes more outliers than existing depth map fusion methods. Even for large outlier fractions, our method successfully filters almost all of them.

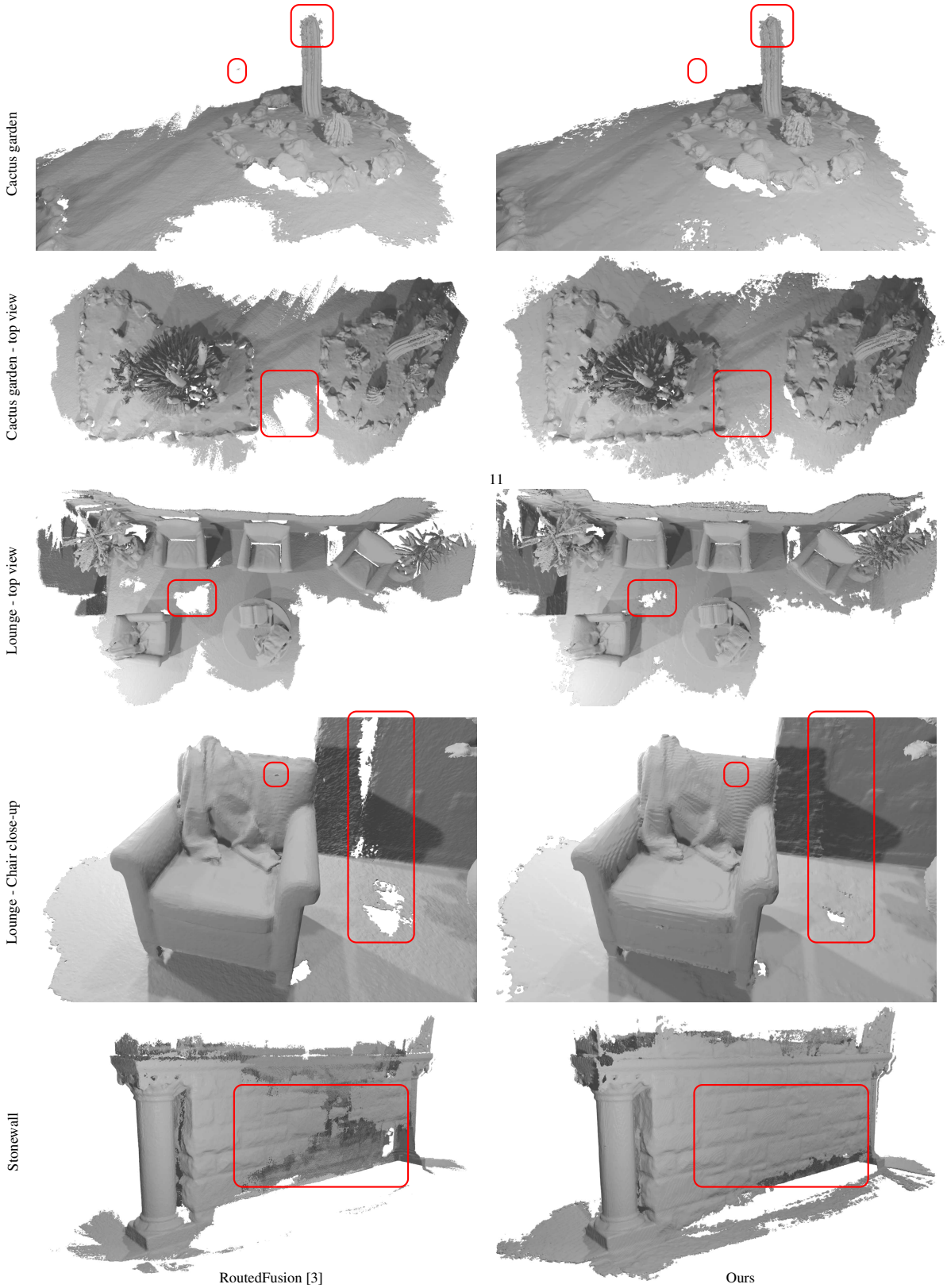

Figure 3: **Additional results on Scene3D [5]**. Our method reconstructs scenes with significantly higher completeness. This is due to the learned translation that can effectively discriminate between outliers and geometry. Furthermore, our method can filter large outlier blobs.

References

[1] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, pages 303–312, 1996.

[2] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018.

[3] Silvan Weder, Johannes L. Schönberger, Marc Pollefeys, and Martin R. Oswald. RoutedFusion: Learning real-time depth map fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[4] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1912–1920. IEEE Computer Society, 2015.

[5] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Trans. Graph.*, 32(4):112:1–112:8, 2013.