Learning Online Multi-Sensor Depth Fusion Supplementary Material

Erik Sandström[1], Martin R. Oswald[1,2], Suryansh Kumar[1], Silvan Weder[1], Fisher Yu[1], Cristian Sminchisescu[3,5], and Luc Van Gool[1,4]

[1]ETH Zürich, [2]University of Amsterdam, [3]Lund University, [4]KU Leuven, [5]Google Research

Abstract. This supplementary material accompanies the main paper by providing further information for better reproducibility as well as additional evaluations and qualitative results.

A. Videos

We provide an introductory video that presents an overview of our method (available here: `https://youtu.be/woA8FU05AM0`) as well as a summary of the most important results. Additionally, a selection of short videos is linked in the description of the introductory video showing the online reconstruction process for various sensors and scenes.

B. Method

In the following, we provide more details about our proposed method, specifically the Shape Integration Module which updates the shape grid S_t^i.

Shape Integration Module. The shape integration module takes as input a depth map D_t^i from sensor $i \in \{0, 1\}$ at time $t \in \mathbb{N}$, with known camera calibration $P_t^{c\to w} \in \mathbb{SE}(3)$ from camera to world space and intrinsics $K_t \in \mathbb{R}^{3\times 3}$ and performs a full perspective unprojection to attain a point cloud $\mathbf{X}_w$ in the world coordinate space. Each 3D point $\mathbf{x}_w \in \mathbf{X}_w$ is computed by transforming each pixel $(u,\ v)$ of the depth map into the camera space $\mathbf{x}_c$ according to (1),

$$\mathbf{x}_c = D_t^i(u, v) K_t^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{1}$$

and then into the world camera space according to (2).

$$\mathbf{x}_w = P^{c\to w} \begin{bmatrix} \mathbf{x}_c \\ 1 \end{bmatrix} \tag{2}$$

Along each ray from the camera center, centered at $\mathbf{x}_w$, we sample T points evenly over a predetermined distance l. The distance l and the number of points

T are selected based on the noise level of the depth sensor and could be interpreted similarly as the truncation distance in standard TSDF Fusion [5]. The distance between sampled points should ideally be the same as the voxel side length. After computing T points along each ray, we convert the coordinates to the voxel space and extract a local shape grid $S^{i,*}_{t-1}$ from S^{i}_{t-1} through nearest neighbor extraction. To incrementally update $S^{i,*}_{t-1}$, we follow the moving average update scheme of TSDF Fusion,

$$V^{i,*}_{t} = \frac{W^{i,*}_{t-1} V^{i,*}_{t-1} + v^{i}_{t}}{W^{i,*}_{t-1} + 1}, \tag{3}$$

where v^{i}_{t} is the TSDF update. The local weight grid $W^{i,*}_{t}$ is updated as

$$W^{i,*}_{t} = \max(\omega_{\max}, W^{i,*}_{t-1} + 1). \tag{4}$$

The weights are clipped at $\omega_{\max}$ to prevent numerical instabilities.

Denoising Network. We use the identical denoising network as described by Weder *et al.* [14] (called routing network) except that we change the loss hyperparameter to $\lambda = 0.06$.

Indicator Function. We define the indicator function $\mathbb{1}_{\{A\}}(x)$ for a voxel index $x = (x_1,\ x_2,\ x_3)$, where $x_i \in \mathbb{N}$ as

$$\mathbb{1}_{\{A\}}(x) = \begin{cases} 0 & \text{if } x \notin A \\ 1 & \text{if } x \in A, \end{cases} \tag{5}$$

and for two sets A and B

$$\mathbb{1}_{\{A,\ B\}}(x) = \begin{cases} 0 & \text{if } x \notin A \cap B \\ 1 & \text{if } x \in A \cap B. \end{cases} \tag{6}$$

In the main paper, we omit x for brevity.

C. Implementation Details

We use PyTorch 1.7.1 and Python 3.8.5 to implement the pipeline. Training is done with the Adam optimizer using an Nvidia TITAN RTX with 24 GB of GPU memory. We use a learning rate of 1e-04 and otherwise the default Adam hyperparameters $betas = (0.9, 0.999)$, eps = 1e-08 and $weight\_decay = 0$. We use a batch size of 1 due to the online nature of the pipeline, but accumulate the gradients over 20 frames before updating all network weights. We shuffle the frames during train time and for training efficiency, we integrate every 10th frame during validation. We record a runtime of ∼15 fps on our unoptimized implementation on the Human CoRBS scene. For our largest scenes (*e.g. Hotel 0*), our integration frame rate is between 1-2 fps. The bottleneck is cpu-gpu communication where our implementation loads the full voxel grid to the gpu for fast updates and then loads the grid back to the cpu to allow for scene reconstruction of multiple scenes in parallel. We train our network until convergence which takes between 12-24 hours on the Replica dataset and a few hours on the CoRBS and Scene3D datasets.

D. Evaluation Metrics

We use the following seven metrics to quantify the reconstruction performance. **Voxel Grid Metrics.** We use four metrics on the TSDF voxel grid. We mask the evaluation so that only voxels with a non-zero weight W_k are considered. Mean Absolute Distance (MAD): Computed as the mean of the L_1 error to the ground truth signed distance grid MAD= $\frac{1}{N}\sum_{k=0}^{N}|V_k - V_k^{GT}|_1$, where N is the total number of valid voxels. Mean Squared Error (MSE): Computed as the mean squared error to the ground truth signed distance grid MSE= $\frac{1}{N}\sum_{k=0}^{N}(V_k - V_k^{GT})^2$, where N is the total number of valid voxels. Intersection over Union (IoU): Computed on the occupancy grid of the voxel grid as IoU= $\frac{tn}{tn+fp+fn}$ and Accuracy as Acc= $\frac{tn+tp}{tp+tn+fp+fn}$, where

$$tn = \sum\{\text{sign}(V) < 0 \text{ and } \text{sign}(V^{GT}) < 0\} \tag{7}$$

$$tp = \sum\{\text{sign}(V) >= 0 \text{ and } \text{sign}(V^{GT}) >= 0\} \tag{8}$$

$$fp = \sum\{\text{sign}(V) >= 0 \text{ and } \text{sign}(V^{GT}) < 0\} \tag{9}$$

$$fn = \sum\{\text{sign}(V) < 0 \text{ and } \text{sign}(V^{GT}) >= 0\} \tag{10}$$

Mesh Metrics. We run marching cubes [11] on the predicted voxel grid V and the ground truth voxel grid V^{GT} and compare the two meshes. The F-score is defined as the harmonic mean between Recall (R) and Precision (P), $F = 2\frac{PR}{P+R}$. Precision is defined as the percentage of vertices on the predicted mesh V_m which lie within some distance τ from a vertex on the ground truth mesh V_m^{GT}. Vice versa, Recall is defined as the percentage of points on the ground truth mesh V_m^{GT} which lie within the same distance τ from a vertex on the predicted mesh V_m. In all our experiments, we use a distance threshold $\tau = 0.02$ m. We use the provided evaluation script of the Tanks and Temples dataset [10], but modify it to our needs. For a more accurate evaluation, we do not downsample or crop the meshes and we do not utilize the automatic alignment procedure since our meshes are already aligned.

E. Replica Dataset Collection

Due to the lack of available data for the study of multi-sensor depth fusion, we construct our own 2D dataset from the 3D Replica dataset [13], which comprises 18 high-quality scenes. To compute the ground truth signed distance value at each voxel grid point, we require a well-defined normal direction. This is not provided by the non-watertight Replica meshes. Thus, we apply screened Poisson surface reconstruction with CloudCompare [6], with an octree depth of 12. Otherwise, the default settings are used. Additionally, we found that the Poisson surface reconstructions are not clean enough to produce high-quality signed distance grids. Thus, each watertight mesh is cleaned with the Meshlab [4] filter function `remove isolated pieces with respect to face number`. We

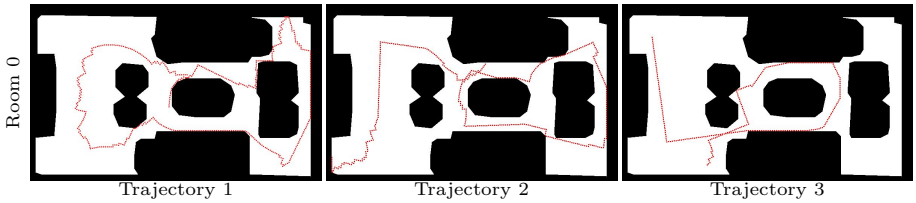

Fig. 1: **Trajectory Visualization.** Top-down visualization of the three manually traversed trajectories for the *room 0* scene. Navigable space is colored white.

set the face number threshold to 100. The signed distance grids are then computed from the meshes with a modified version of the mesh-to-sdf library[1] to accommodate non-cubic voxel grids.

Using the Habitat AI platform [12], we define an agent that moves around in the watertight 3D scenes by traversing the scenes manually using the keyboard. We sample three trajectories per scene with diverse starting points and capture the scene content to simulate a realistic capturing scenario *i.e.* for instance a human moving a mobile device. For each step that the agent takes, it moves 0.05 m along the x- or y-axis. When the agent rotates, it rotates 2.5 degrees. The step sizes were chosen so that the dataset also can be used to evaluate multi-sensor tracking methods in the future. The agent is equipped with two pairs of RGBD cameras. Both cameras are located at a fixed height of 1.5 m above the floor and the baseline between the cameras is 0.1 m. We use an identical resolution of 512×512 for both cameras and a field of view of 90 degrees. The full dataset comprises 92698 frames, of which we use a subset to produce a training, validation and testing set. For example, a dense voxel grid of the *apartment* 0 scene did not fit on the GPU, and the variations of the *frl apartment* scenes did not provide any further diversity in the data. The train set consists of the scenes {*apartment 1, frl apartment 0, office 1, room 2, office 3, room 0*} and contains 22358 frames, the validation set consists of the scene {*frl apartment 1*} and contains 1958 frames and the test set consists of the scenes {*office 0, office 4, hotel 0*} and contains 1891 frames. When training our model, we use all three trajectories from each train scene. During validation, only trajectory 1 is used. During testing, we use trajectory 3 for *hotel 0* and *office 4* and trajectory 1 for *office 0*. As an example, in Fig. 1, we visualize a top-down view of the three manually traversed trajectories for the *room 0* scene. The trajectory is visualized in red and the navigable floor is white.

F. Scene3D Dataset

The Scene3D dataset comprises multiple scenes, but we only evaluate the Copy room scene. We found that the ground truth meshes of all other scenes were not complete enough, which made the evaluation inaccurate.

[1] `https://github.com/marian42/mesh_to_sdf`

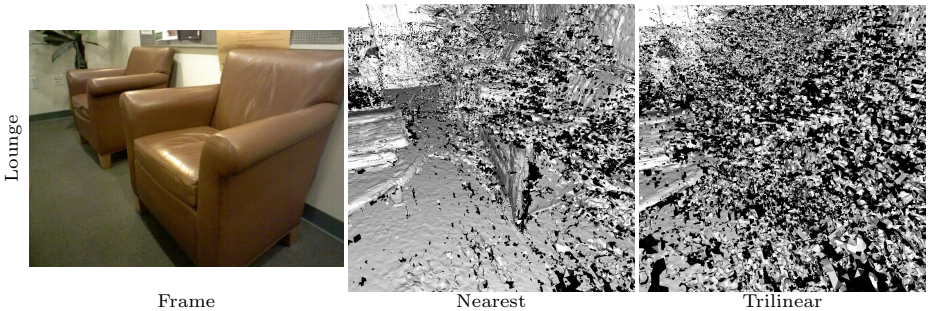

Fig. 2: **RoutedFusion Masking.** The trilinear interpolation mask of RoutedFusion [14] results in significantly more outliers than using the nearest neighbor mask. Trilinear interpolation updates eight grid points per sampled point along the ray instead of two. This exposes more outliers as marching cubes [11] requires that all eight grid points have a non-zero weight for a surface to be drawn.

G. Baselines

RoutedFusion. We found that the original implementation of RoutedFusion generates significantly more outliers than TSDF Fusion [5] when no weight thresholding is applied as post-outlier filter. This is due to the trilinear interpolation extraction step of RoutedFusion compared to the nearest neighbor extraction of TSDF Fusion. Trilinear interpolation updates eight grid points per sampled point along the ray instead of one. This exposes more outliers as marching cubes [11] requires that all eight grid points have a non-zero weight for a surface to be drawn. Fig. 2 compares the meshes of two masks (the sets of non-zero weights) on the same RoutedFusion model. The trilinear interpolation mask is the standard output of RoutedFusion while the nearest neighbor mask is taken from running TSDF Fusion on the same scene. Given the significantly better result with the nearest neighbor mask, we report all results in the paper using the nearest neighbor mask.

Early Fusion. We use the denoising network described by Weder *et al.*[14] (called routing network) as basis for our Early Fusion baseline. We increase the number of input channels by one so that the sensor depth maps can be fused and we use the loss hyperparameter $\lambda = 0.06$.

DI-Fusion. For a fair comparison between all models, we make the following modifications to the original implementation of DI-Fusion [9]. 1) We turn off the camera tracker and provide the ground truth camera poses. 2) We turn off the heuristic pre-outlier filter used on the incoming depth maps. 3) We turn off the heuristic weight counter thresholding post-outlier filter. 4) We use a voxel size of 2 cm, but sample the grid at a simulated resolution of 1 cm. This is achieved by setting the `resolution` variable in the config file to 2. We note that the implementation does not support `resolution < 2`. Furthermore, the implementation does not allow for convenient access the dense voxel grid and thus, we only report the mesh metrics.

H. Depth Sensor Details

Synthesized ToF Depth. The noise model[2] [7] incorporates disparity-based quantization, high-frequency noise, and a model of low-frequency distortion estimated on a real depth camera. We increase the noise level of the depth by increasing the standard deviation of the high-frequency noise by a factor of 5. We also multiply the standard deviation of the pixel shuffling with the same factor. Other works have previously used this model for the evaluation of dense surface reconstruction methods [3, 15, 16].
PSMNet Stereo Depth. We first pretrain PSMNet [2] on the SceneFlow dataset according to the documentation provided by Chang *et al.*. The model is then fine-tuned on our Replica dataset. During training we use the default parameters.
SGM Stereo Depth. We generate depth maps with Semi-Global Matching [8] implemented in OpenCV [1]. We set the number of disparities $numDisparities = 64$ and use the full variant of the algorithm which considers 8 directions instead of 5 by setting $mode = MODE\_HH$. Otherwise, we use the default settings.
COLMAP MVS Depth. Dense depth maps are computed with known camera poses and intrinsics using the sequential matcher with a 10 image overlap.

I. More Experiments

Time Asynchronous Evaluation. RGB cameras often have higher frame rates than ToF sensors which makes Early Fusion more challenging as one sensor might lack new data. Tab. 2 in the main paper provides performance results when the ToF sensor has half the sampling rate compared to the PSMNet sensor. In Tab.1, we show that the performance gap to our method grows even more when the sampling rate is decreased to a third (of the PSMNet sensor) for the ToF sensor. The drop in performance compared to our method can be attributed to the reprojection of the ToF frames. The reprojection step introduces occlusions and pixel quantization errors. The performance of our method actually slightly improves on some metrics. This can be explained by the fact that the completeness of the scene is saturated and dropping ToF frames removes noise and outliers that would otherwise have been integrated. See also Fig.6 for a visualization. We do not retrain any model for this experiment.
Performance over Camera Trajectory. To show that our fused output is not only better at the end of the fusion process, we visualize the quantitative performance across the accumulated trajectory for the *office 0* scene for the model {ToF, PSMNet} with denoising in Fig. 3. Our fused model consistently improves on the inputs. Note that we only show the metrics IoU and MAD in the main paper.

[2] `http://redwood-data.org/indoor/dataset.html`

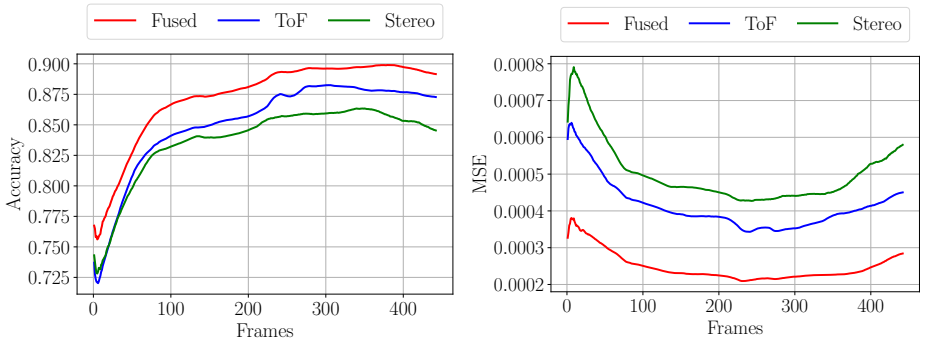

Fig. 3: **Performance over Camera Trajectory.** The fused output of our method outperforms the single-input reconstructions for all frames along the trajectory. The experiment was done on the *office 0* scene for the sensors {ToF, PSMNet} with denoising. Note that the results get slightly worse after 300 frames. This is due to additional noise from the depth sensors when viewing the scene from further away.

Sampling Rate ToF	Metric / Model	MSE↓ *e-04	MAD↓ *e-02	IoU↑ [0,1]	Acc.↑ [%]	F↑ %	P↑ [%]	R↑ [%]
1/2	Early Fusion	7.66	1.99	0.642	84.65	61.34	48.47	**83.63**
1/3	Early Fusion	8.52	2.15	0.610	83.60	55.52	41.63	83.40
1/2	SenFuNet (Ours)	4.21	1.45	**0.755**	**88.26**	73.04	69.13	78.43
1/3	SenFuNet (Ours)	**4.00**	**1.41**	0.751	88.14	**74.93**	**73.57**	77.05

Table 1: **Time Asynchronous Evaluation.** The gap between SenFuNet and Early Fusion increases further when the ToF camera has a sampling rate of one third compared to the PSMNet sensor.

Weight Ablation. In Tab. 2 we evaluate our full model against two models which only use the weight (W_t^i) as input to the Weighting Network $\mathcal{G}$. We perform {SGM, PSMNet} fusion without denoising on the Replica dataset. We observe that our full model provides a gain on the model which only uses the tanh-transformed weights. This justifies our Feature Network. We also show that a normalization of the weight with a tanh-transformation improves performance over no normalization.

Architecture Ablation. For the architecture ablations, we perform {SGM, PSMNet} fusion without denoising on the Replica dataset. For all experiments, unless otherwise specified, we use 4 layers of 3D-convolutions with kernel size 3 in $\mathcal{G}$, 6 network blocks in the Feature Networks $\mathcal{F}^i$, and store feature vectors of dimension $n = 5$ in the feature grids F_t^i.

In Tab. 3, we investigate the effect of using different number of network blocks in the feature network. Performance is maximized when using 5 blocks.

In Tab. 4 we vary the number of feature dimensions that is stored in the grids. 4 dimensions yield optimal performance.

Finally, in Tab. 5, we study the importance of the Feature Network by testing it against a model which bypasses the network and hence unprojects the 2D features without any 2D processing. For this experiment, both models use 4 feature dimensions to make the comparison fair. Note that no weights were used as input to the Weighting Network. We gain performance by using the feature network, which justifies our design choice.

The ablations suggest that the best performance is achieved with $n = 4$, the number of blocks in the feature networks is 5 and when we use 2 3D-convolutional

Model	F↑ [%]
Only Weights	66.69
Only *tanh*(Weights)	68.92
Full Model	**69.83**

Table 2: **Weight Counter Ablation.** Our full model outperforms models which only use the weight (W_t^i) as input to the Weighting Network $\mathcal{G}$. Normalization of the weight with a *tanh*-transformation improves performance over no normalization.

Nbr Blocks	1	2	3	4	5	6
F↑ [%]	67.45	67.68	66.79	67.59	**68.39**	68.21

Table 3: **Ablation Study.** We change the number of blocks for the feature network. Optimal performance is achieved with 5 blocks.

n	2	3	4	5
F↑ [%]	68.86	68.20	**68.87**	68.21

Table 4: **Ablation Study.** We alter the dimension of the feature vector. The performance is maximized at $n = 4$.

Model	F↑ [%]
Without Feature Net	67.44
With Feature Net	**68.87**

Table 5: **Architecture Ablation.** We demonstrate the difference in performance with and without the feature network.

layers with kernel size 3. Empirically, we found that $n = 5$, 6 feature network blocks and 2 kernel size 3 3D convolutions gave marginally better results and this is what we report throughout the paper.

Effect of Weight Thresholding. RoutedFusion [14] applies weight thresholding to filter outliers *i.e.* a TSDF surface observation needs to have a weight larger than some threshold to not be removed during post-processing. Weight thresholding was not applied to any model in the main paper to avoid the need of tuning an additional parameter and to make the comparison fair between the models. For example, on the Replica scenes, the optimal weight threshold is typically within the range 1-10, while for the CoRBS human scene it is around 500. Tab. 7 shows the effect when weight thresholding is applied on the Replica test set on the sensors {ToF, PSMNet} with denoising. Regardless of the weight threshold, our method outperforms RoutedFusion.

DI-Fusion Ablation. Tab. 6 shows the performance of DI-Fusion [9] for $\sigma = \{0.15, 0.06\}$ compared to our method. The results when $\sigma = 0.15$ is reported in the main paper and this number is also specified in the implementation provided by the authors. $\sigma = 0.06$ is suggested for one experiment in the DI-Fusion paper. SenFuNet outperforms DI-Fusion for both choices of the σ threshold. In general, a high σ yields high recall, but poor precision and vice versa. This is, however, not true for the Human scene where SenFuNet outperforms DI-Fusion on all metrics. On the copyroom scene, SenFuNet outperforms DI-Fusion both in terms of the F-score and precision. On the Replica dataset, SenFuNet attains around 10 percent points higher F-score compared to the best DI-Fusion model.

ToF+ToF denoising Fusion. From the experiments with and without depth denoising in the main paper, we note that the depth denoising network brings advantages where planar noise is present, but disadvantages due to over-smoothing around depth discontinuities. A natural question arises: Can our framework com-

Model	F↑ [%]	P↑ [%]	R↑ [%]	F↑ [%]	P↑ [%]	R↑ [%]
ToF+MVS	*Copyroom*			*Human*		
DI-Fusion [9] σ=0.15	86.31	77.27	**97.74**	28.19	16.71	90.15
DI-Fusion [9] σ=0.06	79.21	91.03	70.11	13.52	18.75	10.56
SenFuNet (Ours)	**93.73**	**91.56**	96.00	**74.56**	**59.74**	**99.16**
Replica: ToF+PSMNet	*w/o denoising*			*w. denoising*		
DI-Fusion [9] σ=0.15	48.39	34.24	**85.29**	55.66	41.49	**85.33**
DI-Fusion [9] σ=0.06	60.30	**72.49**	51.88	63.02	**75.14**	54.46
SenFuNet (Ours)	**69.29**	62.05	79.81	**76.47**	73.58	79.77
Replica: SGM+PSMNet	*w/o denoising*			*w. denoising*		
DI-Fusion [9] σ=0.15	47.29	32.92	**85.14**	52.65	38.50	**83.62**
DI-Fusion [9] σ=0.06	59.24	**71.73**	50.61	63.39	**75.59**	54.73
SenFuNet (Ours)	**69.83**	63.20	79.12	**71.18**	66.81	76.27

Table 6: **DI-Fusion Ablation.** SenFuNet outperforms DI-Fusion for various choices of the σ threshold. In general, a high σ yields high recall, but poor precision and vice versa. This is, however, not true for the Human scene where SenFuNet outperforms DI-Fusion on all metrics. On the Replica dataset, SenFuNet attains around a 10 higher F-score compared to the best DI-Fusion model.

Weight Threshold	1	3	5	7	9	11
RoutedFusion [14] F↑ [%]	62.12	71.40	74.12	75.07	75.11	74.99
Ours F↑ [%]	**77.24**	**79.35**	**79.67**	**79.39**	**78.47**	**77.40**

Table 7: **Weight Thresholding.** Our model outperforms RoutedFusion on the Replica test set also when weight thresholding is applied.

Model \ Metric	MSE↓ *e-04	MAD↓ *e-02	IoU↑ [0,1]	Acc.↑ [%]	F↑ [%]	P↑ [%]	R↑ [%]
Single Sensor							
ToF [7]	7.48	1.99	0.664	83.65	58.52	45.84	**84.85**
ToF denoising [7]	5.08	1.58	0.709	87.32	68.93	59.01	83.08
Multi-Sensor Fusion							
Ours	**3.19**	**1.26**	**0.791**	**90.99**	**78.87**	**76.56**	81.68

Table 8: **ToF+ToF denoising Fusion.** Our method learns to combine the preprocessed depth and the raw depth in a fashion that improves the overall reconstruction.

bine the best of both worlds by fusing a raw ToF frame with a ToF frame preprocessed by a denoising network? Tab. 8 along with Fig. 4 show that our fused output significantly improves on the inputs. For example, the raw ToF contains many outliers behind the walls, while the ToF denoising sensor does not. Vice versa, the raw ToF does not contain outliers around the table and chairs, while the ToF denoising sensor does. As a result, our method selects the appropriate noise-free sensor where needed. Fig. 5 provides the camera trajectory that was used for the evaluation. Note that the raw sensor is favored on surfaces that are viewed close to the camera while the ToF denoising sensor is favored when the measured depth is large.

Visualizations. In Fig. 8 we show qualitative results on the sensors {ToF, PSMNet}. As concluded from the experiment on the sensors {ToF, ToF denoising}, the ToF sensor (without denoising) is not favored when the depth is large. We observe the same prediction on the office 0 scene in Fig. 8. Fig. 7 shows a comparison between our method on the sensors {SGM, PSMNet} without denoising and TSDF Fusion [5], RoutedFusion [14] and DI-Fusion [9]. We achieve better surface reconstruction performance than RoutedFusion and DI-Fusion and better outlier handling than all baseline methods. In

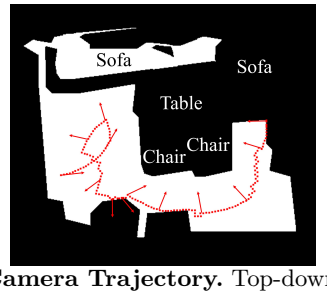

Fig. 5: **Camera Trajectory.** Top-down visualization of the camera trajectory used for the office 0 test scene in Fig. 4. Navigable space is colored white and the arrows show the camera direction.

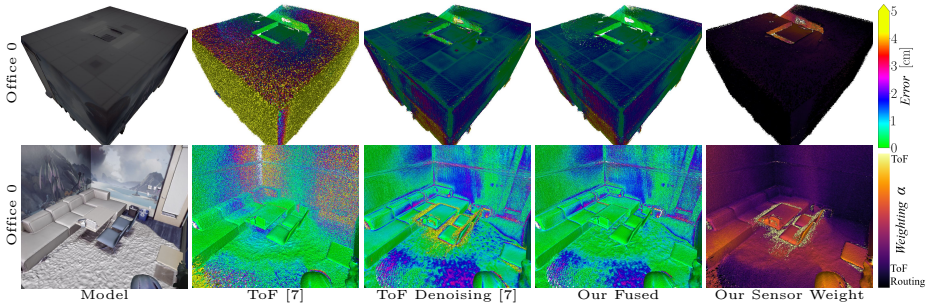

Fig. 4: **ToF+ToF Denoising Fusion.** Our method fuses the raw ToF sensor with the depth denoising preprocessed ToF sensor such that the fused result is improved. Note for example that the outliers from outside the walls from the raw sensor are removed and so are the outliers around the table from the denoising ToF sensor. Fig. 5 provides the camera trajectory that was used for the evaluation. Note that the raw sensor is favored on surfaces that were viewed close to the camera while the denoising ToF sensor is favored when the measured depth is large. See also Tab. 8.

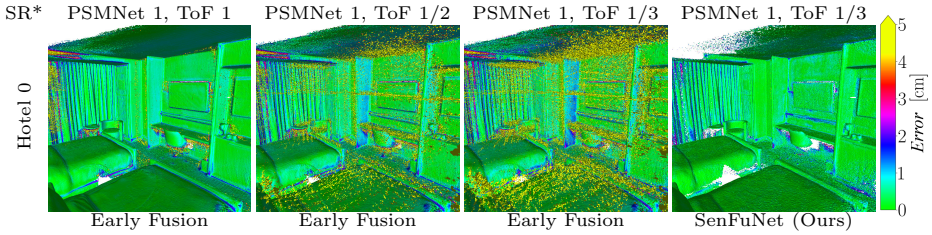

Fig. 6: **ToF+PSMNet Fusion.** The performance gap to our method grows when asynchronous sensors are considered. The performance decreases further for the Early Fusion when the sampling rate is reduced to 1/3 compared to the PSMNet sensor while out method remains robust (best viewed on screen). SR* = Sampling Rate.

Fig. 9, we add depth denoising to the model and evaluate the baseline methods against our method again. Our model achieves better outlier handling overall and more precise surface reconstruction in most regions compared to the Early Fusion method.

For more visual results, we refer to the supplementary videos.

J. Limitations

While our method generates better reconstructions on average, specific local regions may still not improve if the wrong sensor weighting is estimated. Fig. 10 shows four failure cases of our method. The top left visualization of {SGM, PSMNet} without depth denoising shows that the PSMNet surface is selected to a large degree. Our method typically selects the more smooth surface (PSMNet), when compared to a noisy surface (SGM), even though the noisier surface (SGM) may be better on average. The red rectangles on the bottom row and in the top right example show less severe failure cases when our method performs smoothing when selection would have resulted in a more accurate surface

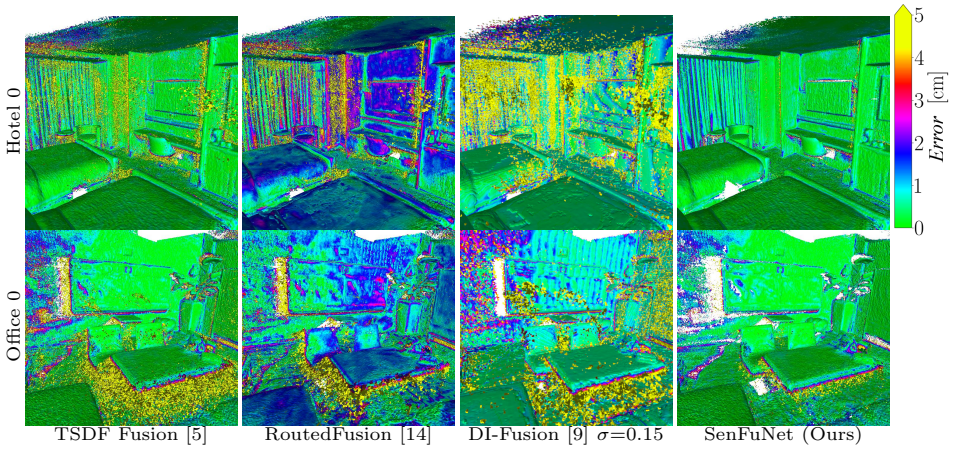

Fig. 7: **SGM+PSMNet Fusion without denoising.** Our method fuses the sensors consistently better than the baseline methods. In particular, our method learns to detect and remove outliers much more effectively (best viewed on screen).

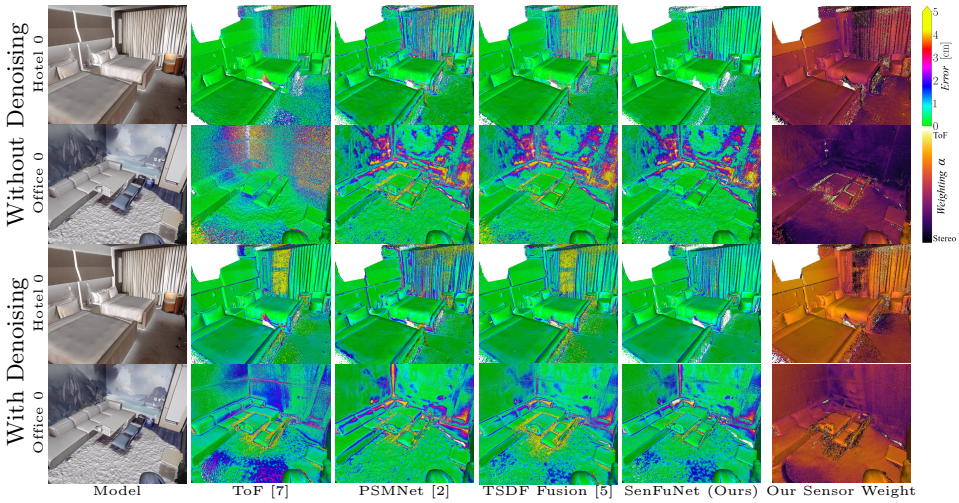

Fig. 8: **ToF+PSMNet Fusion.** Our method fuses the sensors consistently better than TSDF Fusion [5]. In particular, our method learns to detect and remove outliers much more effectively (best viewed on screen).

prediction. This typically happens around edges, but may in rare cases happen on planar regions containing repetitive textures, for example a tiled bathroom shower (see bottom left example). Lastly, our method has difficulties handling overlapping outliers from both sensors *i.e.* where both sensors have registered an outlier at the same voxel. See the orange rectangle in the top right example. This is due to the fact that the Outlier Filter can only be applied on voxels with a single sensor observation.

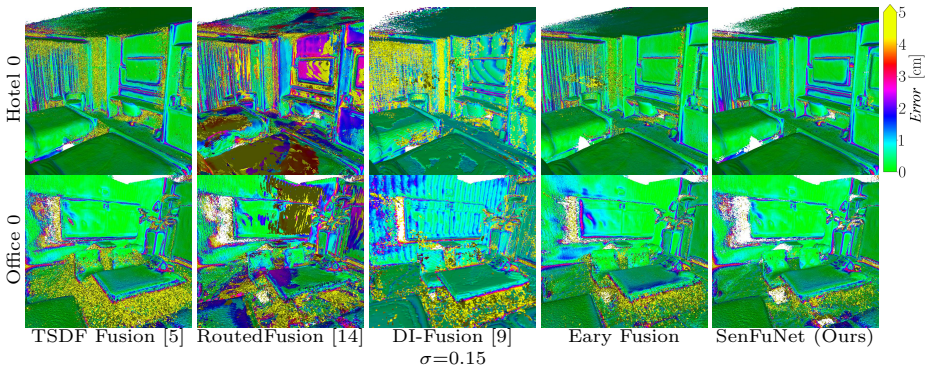

Fig. 9: **SGM+PSMNet Fusion with denoising.** Our method fuses the sensors consistently better than the baseline methods. Compared to the Early Fusion baseline, our method removes more outliers and reconstructs most surfaces better (best viewed on screen).

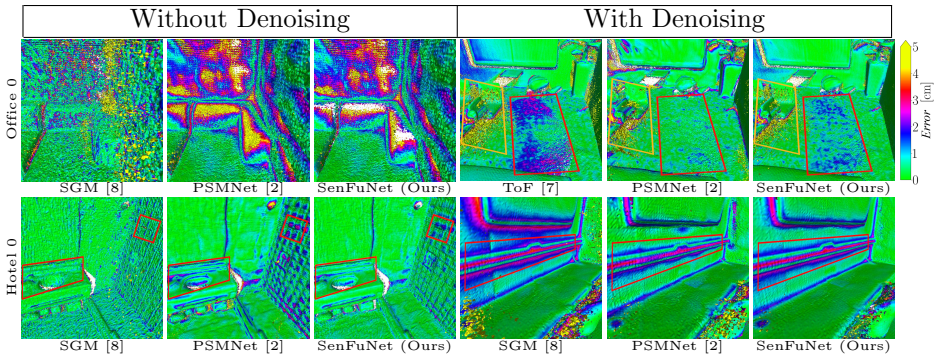

Fig. 10: **Failure Cases.** While our method generates better reconstructions on average, specific local regions may not improve. Overlapping outliers, some edges and cases where one sensor looks noisy but is quantitatively good, are especially difficult to handle (best viewed on screen).

References

1. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
2. Chang, J.R., Chen, Y.S.: Pyramid stereo matching network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5410–5418 (2018)
3. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5556–5565 (2015)
4. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G., et al.: Meshlab: an open-source mesh processing tool. In: Eurographics Italian chapter conference. vol. 2008, pp. 129–136. Salerno, Italy (2008)
5. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 303–312 (1996)
6. Girardeau-Montaut, D.: Cloudcompare. France: EDF R&D Telecom ParisTech (2016)
7. Handa, A., Whelan, T., McDonald, J., Davison, A.J.: A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In: 2014 IEEE international conference on Robotics and automation (ICRA). pp. 1524–1531. IEEE (2014)
8. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Transactions on pattern analysis and machine intelligence **30**(2), 328–341 (2007)
9. Huang, J., Huang, S.S., Song, H., Hu, S.M.: Di-fusion: Online implicit 3d reconstruction with deep priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8932–8941 (2021)
10. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)
11. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
12. Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A Platform for Embodied AI Research. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
13. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., et al.: The replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019)
14. Weder, S., Schönberger, J.L., Pollefeys, M., Oswald, M.R.: Routedfusion: Learning real-time depth map fusion. ArXiv **abs/2001.04388** (2020)
15. Zhou, Q.Y., Koltun, V.: Simultaneous localization and calibration: Self-calibration of consumer depth cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 454–460 (2014)
16. Zhou, Q.Y., Miller, S., Koltun, V.: Elastic fragments for dense scene reconstruction. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 473–480 (2013)