Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction

Supplementary Material

Robert Maier*
maierr@in.tum.de
Raphael Schaller*
schaller@in.tum.de
Daniel Cremers
cremers@in.tum.de

Computer Vision Group
Technical University of Munich
Munich, Germany

* equal contribution

In this document, we provide additional experiments and details of our work "Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction". Specifically, we demonstrate the effect of the different keyframe strategies on the completeness of reconstructions; see Section 1. Moreover, we provide a detailed evaluation of the runtime (Section 2) and memory consumption (Section 3) of our surface correction method. Finally, Section 4 shows some more qualitative results for on-the-fly surface re-integration on additional datasets. Our evaluation was performed on a workstation with Intel Core i7-3770 CPU, 32GB RAM and an NVIDIA GeForce GTX 1070 GPU.

Datasets For our evaluation, we use publicly available RGB-D datasets, an overview is given in Table 1. All used sequences depict larger scenes and provide registered depth and color images (with a resolution of 640×480 at 30 fps) as well as respective camera poses. For assessing the surface quality and to eliminate a substantial source of error, we rely on on the (ground truth) camera poses provided directly with the datasets.

The real-world sequence *TUM/long_office_household* [5] shows a long sequence with a loop closure and provides ground truth camera poses obtained from a high-speed motion capture system.

BundleFusion/apt0 [2] features a very long camera trajectory estimated using a highly accurate 3D reconstruction framework.

AUG_ICL/Liv1 [1] is a synthetic RGB-D sequence generated from manually modeled scenes of a living room. In addition to providing ground truth camera poses, the dataset also provides exact ground truth 3D scene models which allow for a quantitative comparison of surface quality of reconstructed 3D models. The *AUG_ICL* sequences consist of separate sequences with clean and noisy depth maps, which exhibit a more realistic camera noise model.

Surface evaluation methods and metrics In order to perform a quantitative evaluation of our reconstructed 3D models with ground truth models, we first introduce the employed

Sequence	# frames	Synthetic	GT trajectory	GT 3D model
TUM/long_office_household [6]	2486	No	Yes	No
AUG_ICL/Liv1 [1]	2870	Yes	Yes	Yes
BundleFusion/apt0 [2]	8560	No	No	No

Table 1: RGB-D datasets used for our evaluation. (GT stands for ground truth)

methods and metrics. The evaluation procedure for a comparison with synthetic ground truth is adapted from [3] and first extracts a 3D mesh $\mathcal{M}$ from the Signed Distance Field (SDF) volume using the Marching Cubes algorithm [4].

Then, using CLOUDCOMPARE [1], we sample a ground truth reference model $\mathcal{R}$ from the mesh provided for *AUG_ICL/Liv1*, giving a point cloud with 50 million vertices distributed uniformly on the models. Note that we generate distinct models for clean and noisy sequences. By using the poses from the sequences, our models are already aligned with the reference. We use SURFREG [2] to measure the distance of each vertex of our reconstruction $\mathcal{M}$ to its closest vertex in the reference point cloud $\mathcal{R}$ and compute the values for the mean absolute deviation MAD. This technique assesses the *correctness* (CORR) of the model and basically compares the accuracy of the reconstructed surfaces w.r.t. the ground truth model.

However, this method fails to evaluate the *completeness* (COMPL) of the reconstruction, which is especially important for determining the information loss when applying keyframe fusion. For measuring COMPL, we use the inverse procedure and compare every vertex of the reference $\mathcal{R}$ to the nearest neighbor in $\mathcal{M}$. Since the ground truth model contains more surface than actually covered in the synthetic RGB-D frames, we re-generate the reference models by fusing all input frames (without keyframe fusion) into the SDF volume using the ground truth trajectory. This yields reference models that are as complete as possible based on the synthetic frames.

1 Keyframe Strategies

In our work, we have introduced four different strategies for creating Keyframe Fusion keyframes (KF keyframes), namely KF_CONST, KF_DIST, KF_OVRLP and KF_DVO. The number of input RGB-D frames fused into a keyframe has an important effect on both re-integration efficiency and reconstruction surface quality. The more keyframes are generated, the less frames need to be re-integrated on DVO-SLAM pose graph updates, resulting in a more efficient surface correction. At the same time there is a loss in reconstruction quality (completeness COMPL in particular) that goes along with less keyframes, since 3D information cannot be fully represented in the 2.5D keyframe depth maps. Our keyframe fusion generally follows [5] and consists of separate steps for depth and color fusion.

The different keyframe strategies result in a different number of keyframes and consequently directly influence the reconstruction quality. Each keyframe strategy can be configured with specific parameters; the more relaxed these parameters are, the more keyframes are created in general. Figure 1 shows the effect of the different strategies on the completeness COMPL of noisy *AUG_ICL/Liv1*, with parameters chosen in a way that all strategies fuse roughly the same average number $\bar{\kappa}$ of input frames in each keyframe. We additionally show the result of omitting intermediate frames, i.e. only integrating the (unfused) depth map of the input frame corresponding to the keyframe into the SDF volume.

[1] http://www.danielgm.net/cc/
[2] https://github.com/mp3guy/SurfReg

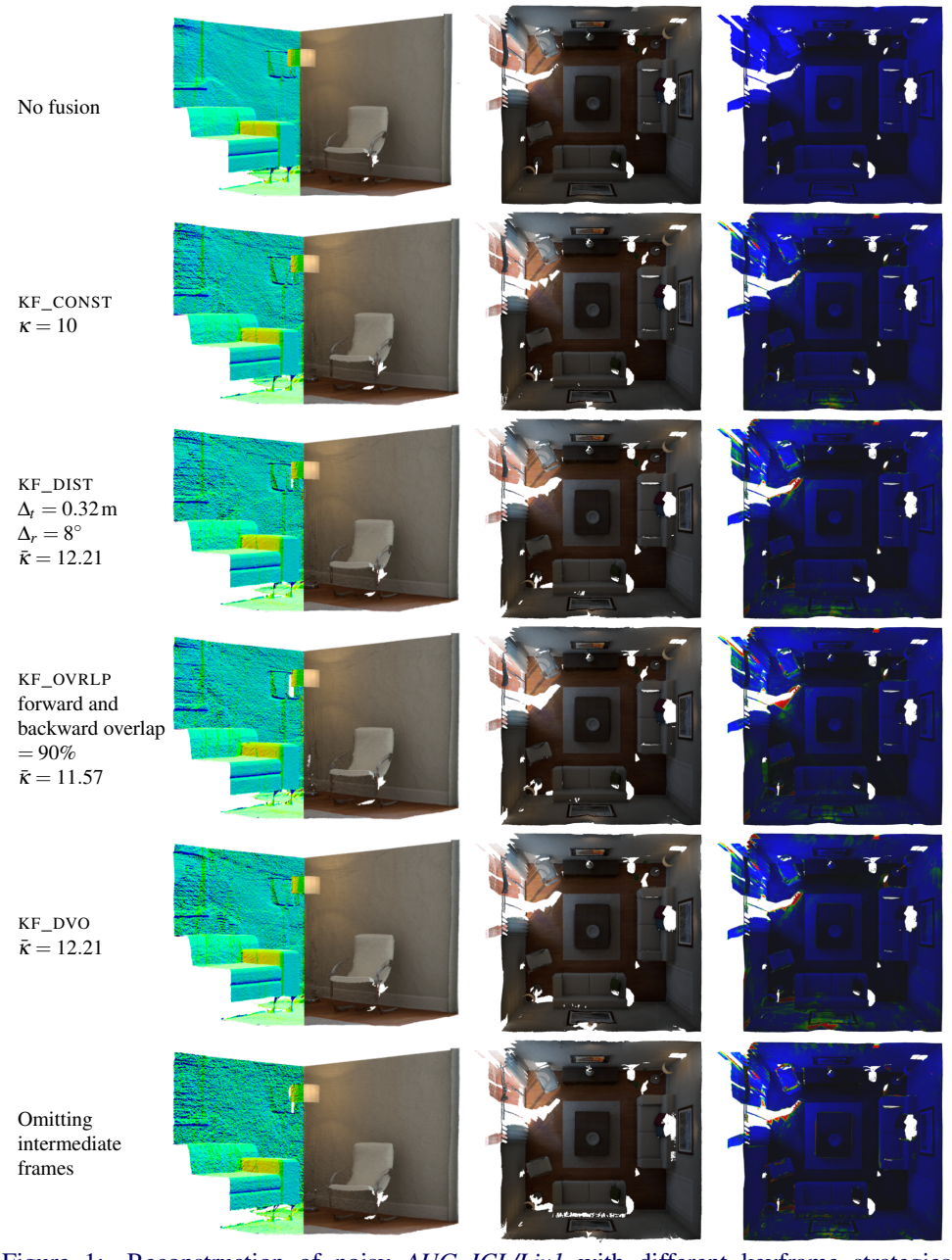

Figure 1: Reconstruction of noisy *AUG_ICL/Liv1* with different keyframe strategies: Keyframe fusion is generally more efficient, but also results in more noisy reconstructions (left) and less complete 3D models (middle and right).

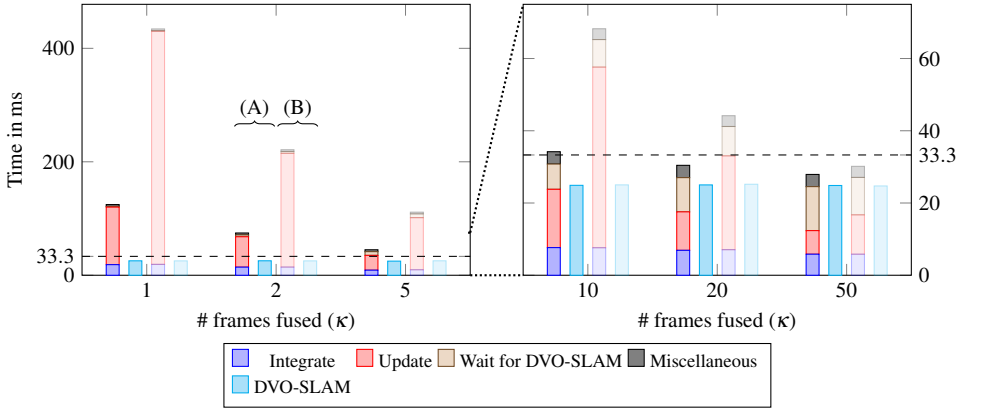

Figure 2: Average runtime required per frame for reconstruction of *AUG_ICL/Liv1* with KF_CONST keyframe strategy w.r.t. the number of frames fused per keyframe κ. For each pose update, $k = 100/\kappa$ frames were re-integrated. Our system (consisting of the steps Integrate, Update, Wait for DVO-SLAM and Miscellaneous, see text) and DVO-SLAM are executed in parallel, hence resulting in the two bars next to each other. While the strong colored bars (A) represent our system run with our improved re-integration strategy, the washed out bars (B) stand for our system run with BundleFusion's strategy. Note that for better visibility of short runtimes, the scale of the figure's x-axis is adapted between $\kappa = 5$ and $\kappa = 10$. The dashed line shows our goal of 30 fps. Table 2 contains the raw data of this figure.

2 Surface Correction Runtime Evaluation

In the following, we show a quantitative runtime evaluation of our method. We rely on the KF_CONST keyframe strategy, since it provices the best trade-off between efficiency, surface quality and predictability (runtime and memory consumption).

Figure 2 gives the average runtimes per frame of our full 3D reconstruction framework w.r.t. the number of input frames fused into a keyframe. Our system is split into it's processing steps:

Integrate Adding of a new frame, i.e., either fusion of a frame into the current keyframe or integration of a keyframe into the 3D model.

Update Correcting the model, i.e., the re-integration procedure.

Wait for DVO-SLAM DVO-SLAM runs in parallel to our system. This component represents the time required for synchronization with DVO-SLAM.

Miscellaneous All other required processing.

For a keyframe size of $\kappa \geq 20$ we achieve real-time performance. DVO-SLAM runs in real-time on a single CPU, asynchronously to our system in a separate thread. Figure 2 also compares our re-integration strategy to BundleFusion's by Dai et al. [2]. Especially for low κ we obtain a significant speed-up stemming from a reduction of runtime required for *Update*.

The raw data of Figure 2 is displayed in Table 2.

Our re-integration strategy					
κ	Integrate	Update	Wait for DVO-SLAM	Miscellaneous	DVO-SLAM
1	18.70	101.40	1.28	3.24	25.45
2	14.31	53.55	3.24	3.39	25.50
5	8.97	26.07	6.55	3.35	24.89
10	7.65	16.14	6.99	3.42	24.89
20	6.90	10.65	9.47	3.41	25.01
50	5.83	6.52	12.18	3.38	24.85

BundleFusion's re-integration strategy					
κ	Integrate	Update	Wait for DVO-SLAM	Miscellaneous	DVO-SLAM
1	19.22	410.83	1.25	2.85	25.32
2	14.42	200.57	3.14	3.06	25.31
5	9.65	91.87	6.68	2.94	25.47
10	7.61	50.01	7.62	2.99	25.01
20	7.07	25.98	8.13	2.98	25.18
50	5.83	10.89	10.37	3.06	24.71

Table 2: Raw data of Figure 2. Our re-integration method requires substantially less time for updating, i.e., correcting, the surface on-the-fly compared to BundleFusion's strategy, while all other steps of the pipeline remain essentially the same.

3 Surface Correction Memory Evaluation

In addition to major runtime improvements during re-integration, keyframe fusion also leads to decreased memory consumption on the host.

Figure 3 shows the memory consumption of our system for different numbers of frames per keyframe κ. The memory consumption was measured every 10^{th} frame and refers to *RSS* (resident set size); we excluded the memory usage of DVO-SLAM, since our surface reconstruction method is independent of the used SLAM system.

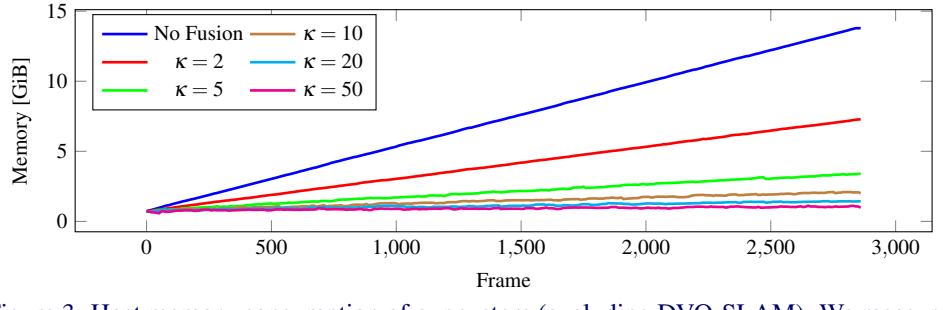

Figure 3: Host memory consumption of our system (excluding DVO-SLAM). We measured the *RSS* (resident set size) after every 10^{th} frame during reconstruction of *AUG_ICL/Liv1* with KF_CONST.

As demonstrated in Figure 3, the *RSS* increases linearly with progressing surface reconstruction (i.e. with the number of integrated frames). Inversely, the memory consumption decreases linearly with increasing κ. In our framework, most of the memory is used for storing the keyframes for potential later re-integration. When comparing *no keyframe fusion* with KF_CONST with $\kappa = 20$, we save about 90% of host memory (14.5 GiB vs. 1.5 GiB).

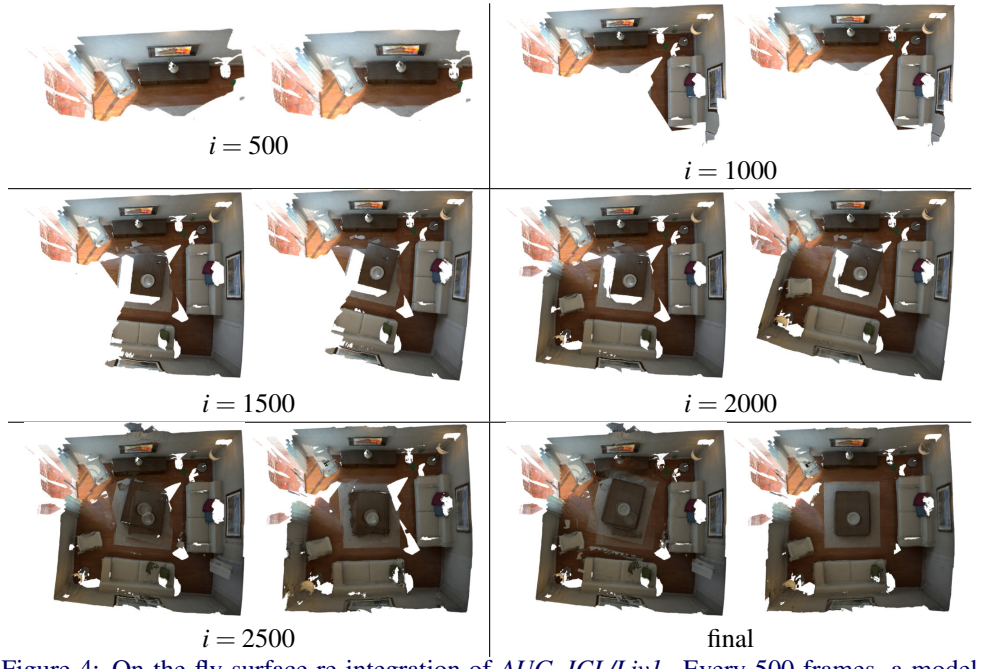

Figure 4: On-the-fly surface re-integration of *AUG_ICL/Liv1*. Every 500 frames, a model was generated.

4 On-the-fly Surface Re-integration

Finally, we present some qualitative examples for on-the-fly surface re-integration on various large-scale datasets.

The following sequences of models were created by reconstructing the RGB-D sequences *TUM/long_office_household*, *BundleFusion/apt0* and *AUG_ICL/Liv1*; the underlying camera poses were provided by DVO-SLAM. With a certain frequency (specified in each figure's caption), a polygon 3D model was generated and later rendered using Blender. In order to compare the outcome with and without re-integration, pairs of renderings are shown, with the left and right image stemming from reconstruction without and with re-integration, respectively. After integration of all frames, the final 3D model was generated, independently of the above frequency. For the run with re-integration, before generation of the final 3D model, *all* frames were re-integrated once to ensure that all pose updates were incorporated in the 3D model.

In particular, Figure 4 shows the results for *AUG_ICL/Liv1*, while Figure 5 shows the model for *TUM/long_office_household* and Figure 6 the reconstruction of *BundleFusion/apt0*.

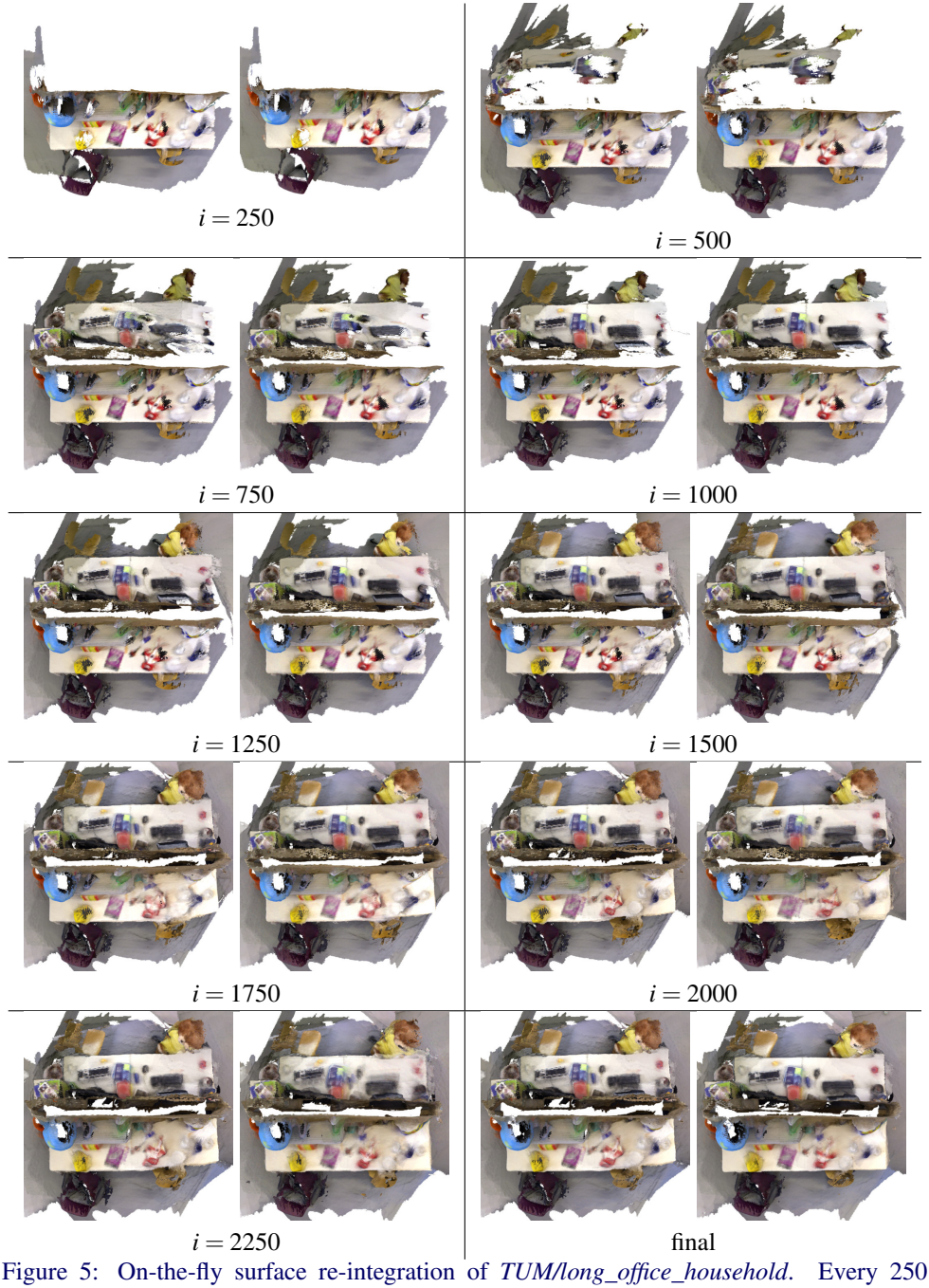

Figure 5: On-the-fly surface re-integration of *TUM/long_office_household*. Every 250 frames, a model was generated.

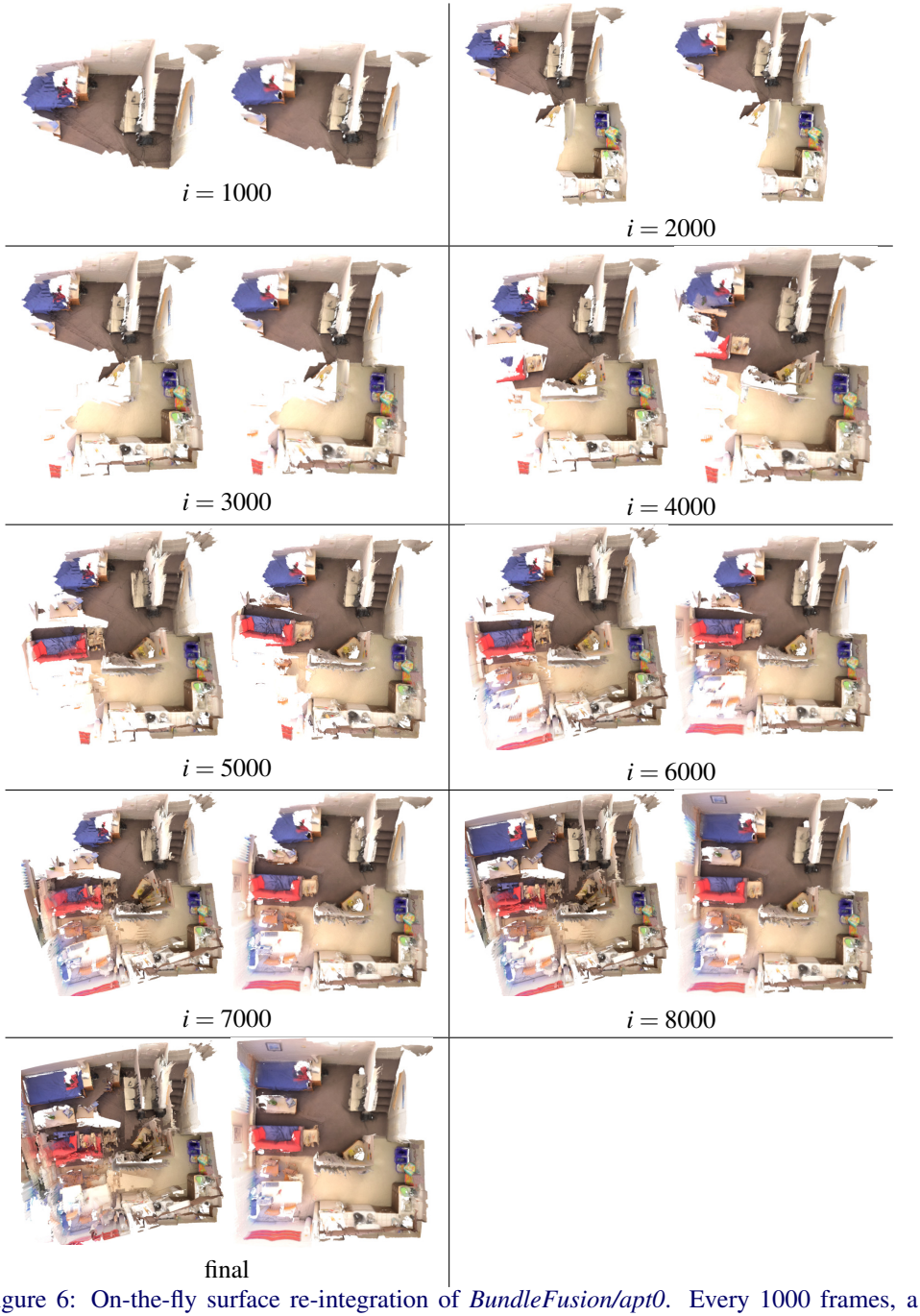

Figure 6: On-the-fly surface re-integration of *BundleFusion/apt0*. Every 1000 frames, a model was generated.

References

[1] S. Choi, Q.Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015.

[2] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics (TOG)*, 2017.

[3] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *ICRA*, 2014.

[4] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987.

[5] R. Maier, J. Stückler, and D. Cremers. Super-resolution keyframe fusion for 3D modeling with high-quality textures. In *3DV*, 2015.

[6] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS*, 2012.