Supplementary Material

1 Running Parameters

As mentioned in Section-VII, the configuration parameters for ORB-SLAM2 is adjusted to reduce the tracking failures as much as possible for each dataset. We mainly lower the threshold for the extraction of ORB features in the provided yaml files, and through the experiments the specific parameters are set as follows:

1) For all sequences on ICL-NUIM datasets, we set the initial threshold as 15 (default: 20), and the minimal threshold as 5 (default: 7).

2) For the *fr3_s_nt_near* and *fr3_s_nt_far* sequences from TUM RGB-D datasets, we set the initial threshold as 12 and the minimal threshold as 3. In addition, we reduce the minimal number of features required for creating an initial sparse feature map from 500 to 200, otherwise the tracking would fail due to the delayed map initialization.

3) For the sub-selected ETH SLAM sequences, we use the value 12 for the initial threshold and 3 for the minimal threshold.

2 Runtime Statistics

Table 1: Average of the median and mean time in [ms] for tracking and mapping on benchmarks datasets. Since our approach is built on top of ORB-SLAM2, we thus take its results (in VO setting) as a baseline. Only the sequences that both methods successfully track are used to computing the average (e.g., *ceiling_2* on ETH_SLAM dataset is excluded). The relative increase of our method compared to ORB SLAM is presented at the bottom.

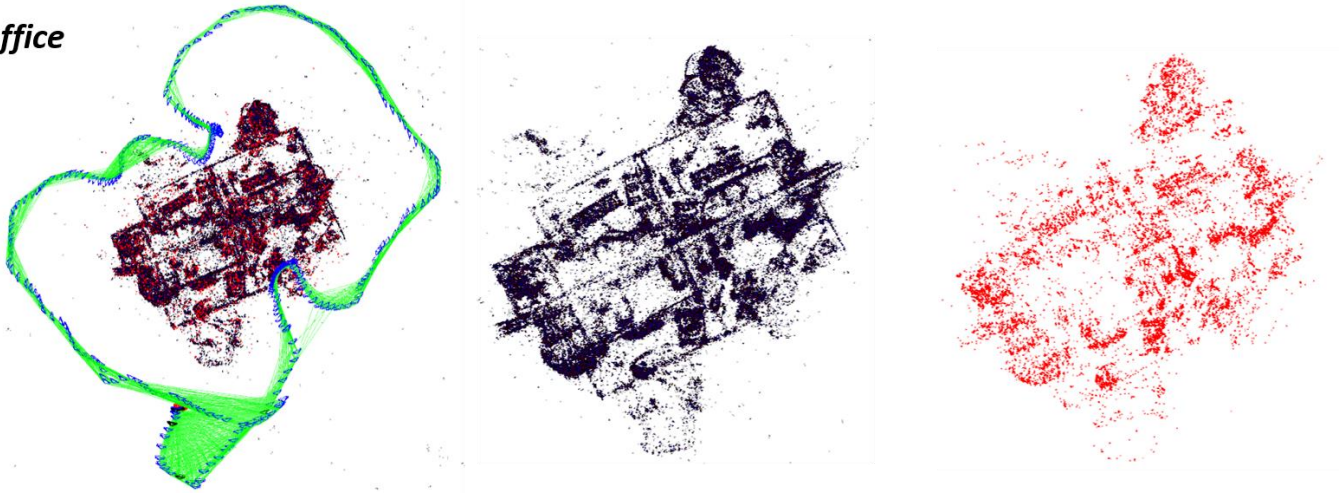
	ICL-NUIM				TUM RGB-D				ETH SLAM			
	Tracking		Mapping		Tracking		Mapping		Tracking		Mapping	
	med.	mean	med.	mean	med.	mean	med.	mean	med.	mean	med.	mean
ORB-VO	20.6	21.2	151	153	25	25.2	222	200	27.6	28.1	172	175
Ours	31.2	32.2	418	402	35.8	36.6	398	396	37.1	38.3	188	220
real_inc.	51.4%	51.8%	176%	162%	43.2%	45.2%	79.2%	98%	34.4%	36.2%	9.3%	25.7%

Although an obvious increase on tracking time can be observed from our method, we think that the absolute tracking time (approx. 35ms in average) can still meet the requirements of some near real-time tasks. On the other hand, our back-end mapping module no surprisingly cost much more time than the original implementation in ORB-SLAM2, except on the ETH SLAM dataset.

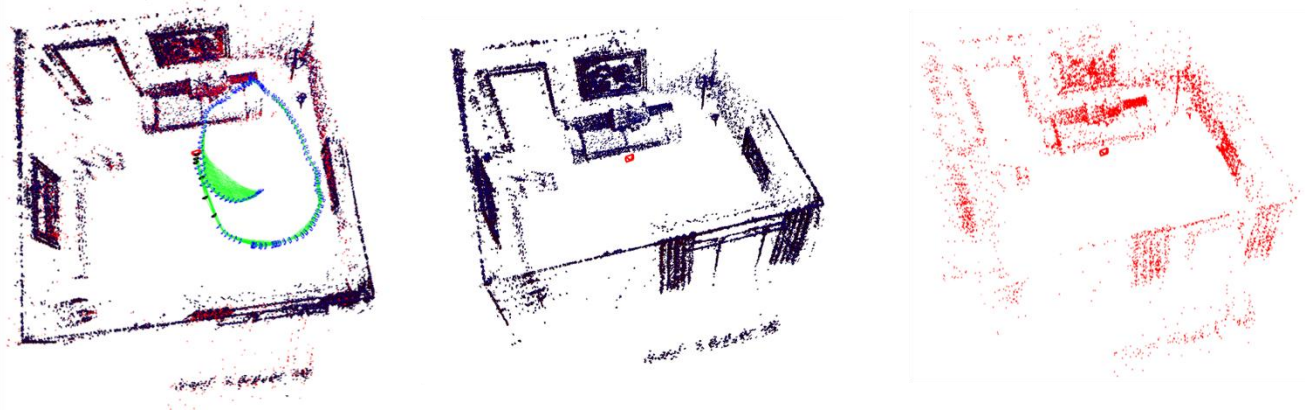
3 Qualitative Examples

The following gives several exampled scenes reconstructed using our VO approach, and from left to right are hybrid maps along with the estimated camera trajectory, the semi-dense point cloud, and the feature-based point cloud. Note that the generated semi-dense geometry matches well with the sparse feature map across synthetic and real sequences (see the zoom-in view).

fr3_office



living_t3_noise



office_t2_noise

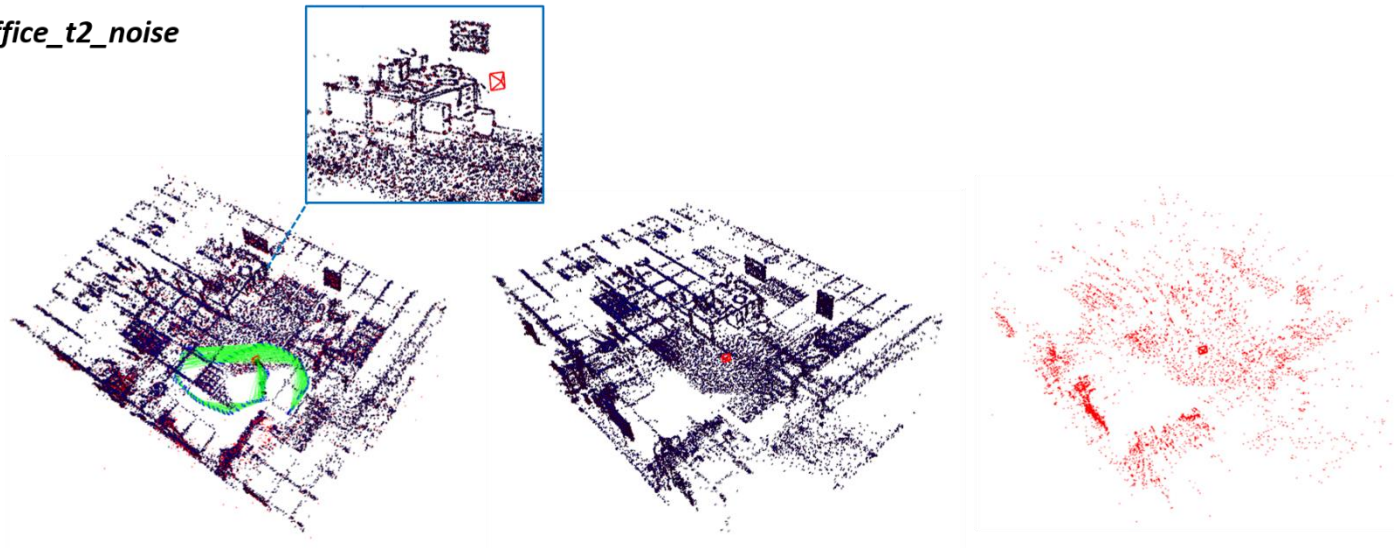
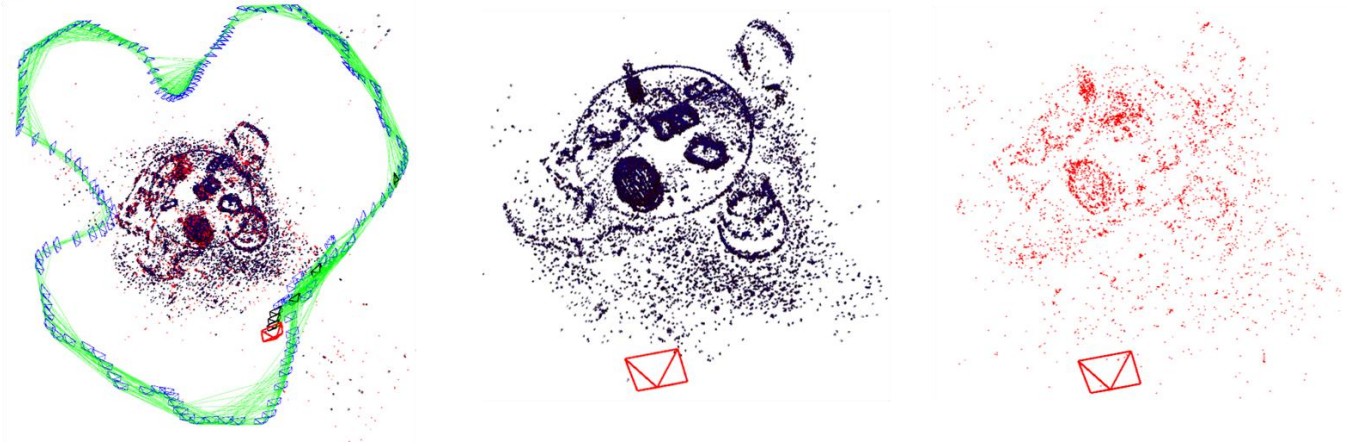
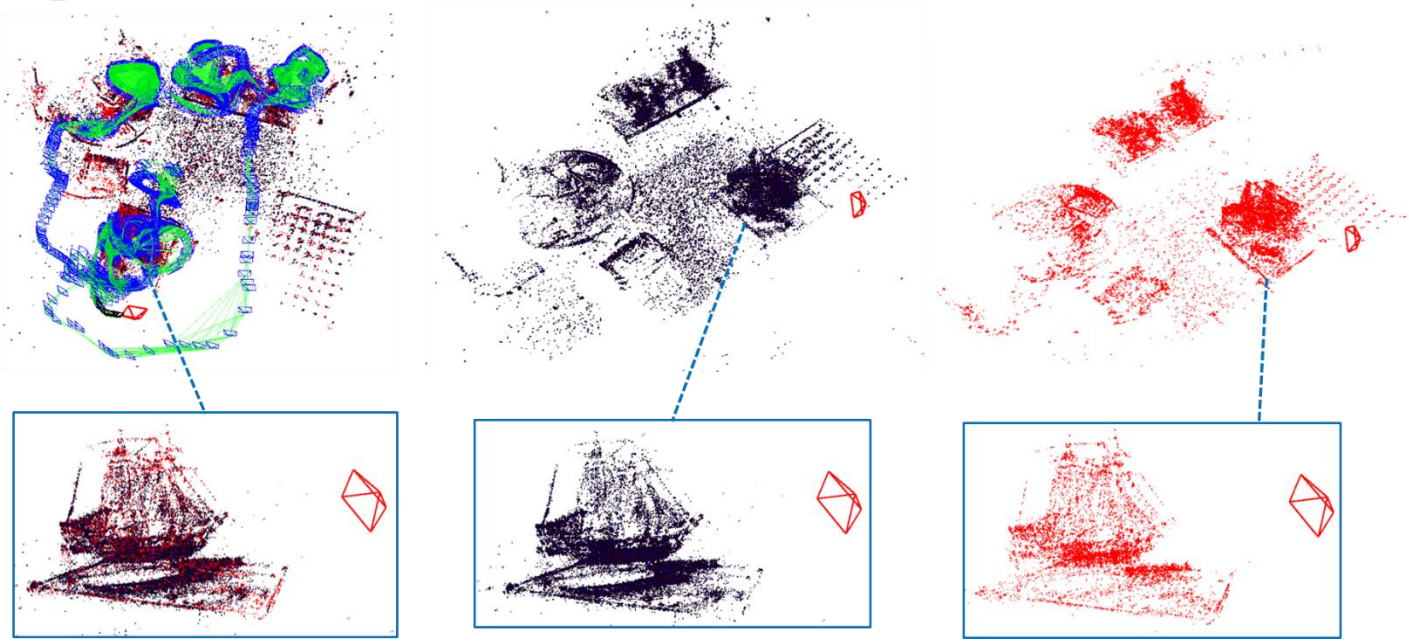


table_3

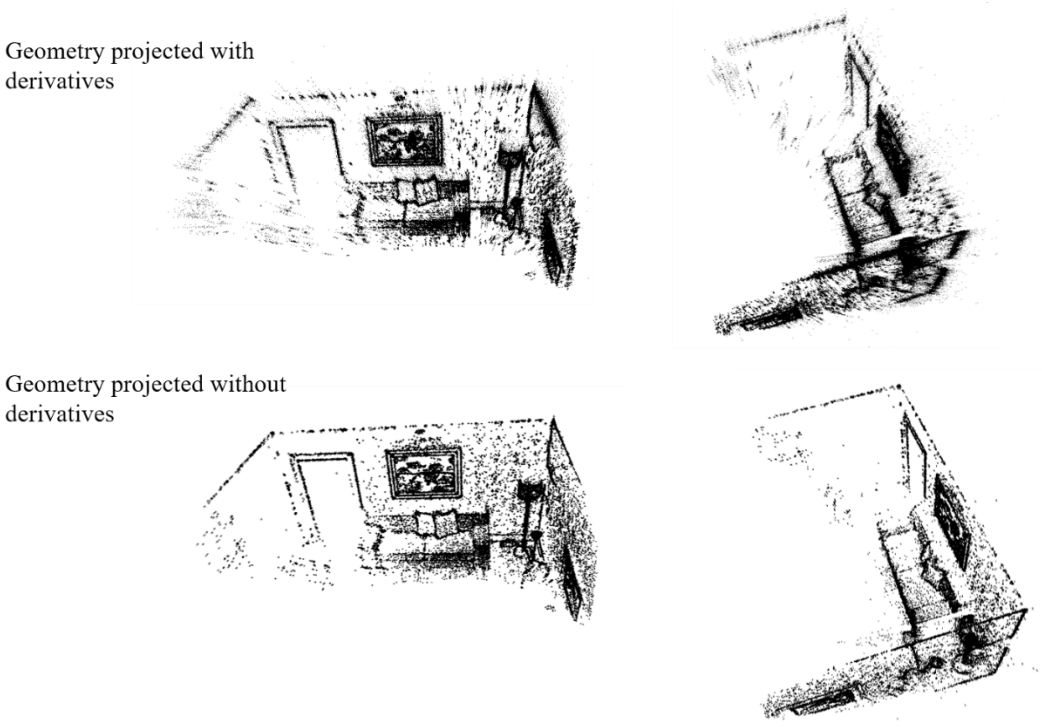


desk_3



4 Usage of Slanted Support Plane Model

It is worth noting that the (inverse) depth derivatives $[d_{u,x},\ d_{u,y}]$ (see Eq. 10) are used as the auxiliary variables to enhance the estimation accuracy of depth d_u associated to the central pixel, and not applied to the direct BA stage for computing the cost term defined in Eq. 7. This means that when computing the cost term over the pixel neighborhood (defined in Eq. 7), this center depth is propagated to its neighboring pixels still following a fronto-parallel assumption like in DSO. The reason is that we found radial structures as well as many outliers would be produced if we use the derivatives estimated via a limited number of iterations to model the real 3D geometry, illustrated as follows:



Actually in standard multi-view reconstruction pipelines, the central depth would be propagated to its adjacent pixel that is then optimized over its own neighborhood, and this process would be repeated several times to obtain consistent point clouds. However, such a scheme is not affordable for a real-time VO/SLAM task, and we thereby only take this slanted plane model to better estimate the depth of the central pixel.