Proceedings of the OAGM Workshop 2018 DOI: 10.3217/978-3-85125-603-1-25

Globally Consistent Dense Real-Time 3D Reconstruction from RGBD Data

Rafael Weilharter[1,2], Fabian Schenk[1], Friedrich Fraundorfer[1]

***Abstract*— In this work, we present a dense 3D reconstruction framework for RGBD data that can handle loop closure and pose updates online. Handling updates online is essential to get a globally consistent 3D reconstruction in real-time. We also introduce fused depth maps for each keyframe that contain the fused depths of all associated frames to greatly increase the speed for model updates. Furthermore, we show how we can use integration and de-integration in a volumetric fusion system to adjust our model to online updated camera poses. We build our system on top of the InfiniTAM framework to generate a model from the semi-dense, keyframe based ORB SLAM2. We extensively evaluate our system on real world and synthetic generated RGBD data regarding tracking accuracy and surface reconstruction.**

I. INTRODUCTION

In recent years, the ubiquity of inexpensive RGBD cameras, pioneered by the Microsoft Kinect, has led to a series of applications for 3D scene reconstruction in areas such as augmented/virtual reality, robotics, gaming and general 3D model estimation. Modern 3D reconstruction systems have to provide a globally consistent model on large scale in real-time. This does not only require a robust camera pose estimation algorithm but also on-the-fly model updates that incorporate loop closures and pose refinements. The robust camera motion estimation process is the main difference between current systems and divides them roughly into three categories.: (i) Iterative closest point (ICP) methods aim to align 3D points but require sufficient 3D structure and a correspondence matching step [7], [10]. Instead of point clouds, (ii) direct methods estimate motion by processing image information directly. Dense [8] and semi-dense [3], [5] variants based on the photo-consistency assumption exist. This makes these approaches especially susceptible to illumination changes and direct methods are typically restricted to small inter-frame motion. (iii) Feature-based methods extract features, match correspondences and estimate motion by minimizing the reprojection error [4], [9]. The extracted features are more robust to illumination changes than the direct methods based on the photo-consistency assumption and are also suitable for larger inter-frame motions.

Regardless of the applied method, many systems rely solely on frame-to-model tracking to estimate the camera movement in real-time and sequentially build a dense 3D model [10], [7]. While such systems are real-time capable they accumulate significant drift over time and usually cannot correct this drift by revisiting the same place (see Fig. 1). To tackle this problem, more advanced Simultaneous Localization and Mapping (SLAM) systems perform loop closure and pose graph optimization to reduce the drift. However, such an approach is computationally very expensive and often only acquires a semi-dense [5] or sparse map [9]. If in addition a dense 3D model is desired, the SLAM system usually relies on expensive hardware, e.g. the Bundle Fusion system [3] only runs on a combination of an NVIDIA GeForce GTX Titan X and a GTX Titan Black.

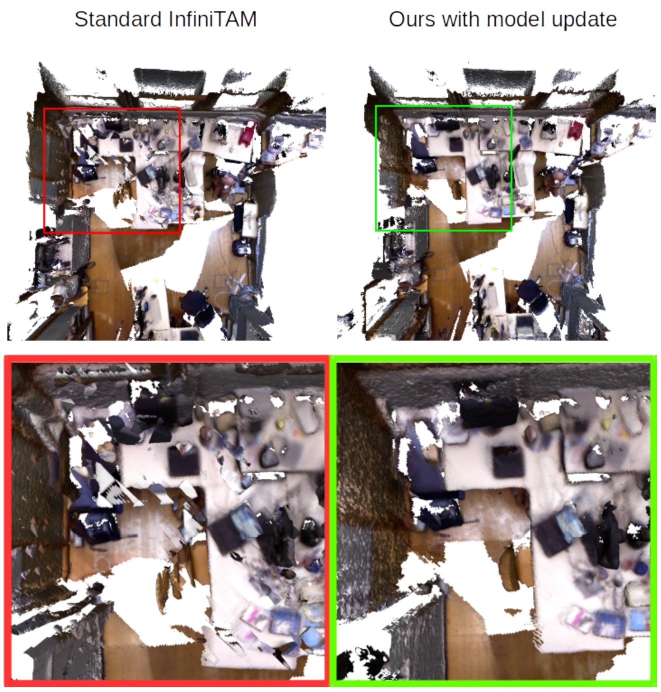


Fig. 1: Reconstructed dense 3D models: No update vs our system with keyframe based depth map fusion and global model update. We can see the effects especially in the upper left corners where a loop closure occurs.

In this paper, we propose a real-time dense 3D reconstruction method that successfully combines the state-of-the-art ORB SLAM2 system [9] with the dense volumetric fusion framework InfiniTAM [7]. To validate our method, we compare the trajectory estimations and surface reconstruction accuracy of several methods on the standard TUM RGBD benchmark dataset and the synthetic ICL NUIM dataset. The

*This work was financed by the KIRAS program (no 850183, CSISmartScan3D) under supervision of the Austrian Research Promotion Agency (FFG).

[1]Institute of Computer Graphics and Vision (ICG), Graz University of Technology, Styria, Austria
[2]Ludwig Boltzmann Institute for Clinical Forensic Imaging (LBI CFI), Graz, Styria, Austria
rafael.weilharter@student.tugraz.at
{schenk, fraundorfer}@icg.tugraz.at

key contributions presented in this work can be summarized as:

- Implementation of a de-integration method which allows to refine and alter the 3D model online when large changes in the estimated trajectory occur, e.g. in the case of a loop closure detection.
- A global model update which can delete and merge keyframes in retrospect.
- A keyframe based depth fusion, where we fuse information of frames into their respective keyframes instead of integrating them directly into the model. This significantly speeds up the re-integration process required for a global model update.
- An extensive evaluation of both, the trajectory error and the surface reconstruction error on several benchmark data sets

II. RELATED WORK

In the past decades, 3D reconstruction has been a very active research field when we focus on work that utilizes RGBD data. Kerl at al. [8] proposed a combination of photometric and geometric error minimization in their visual SLAM system. To reduce the acquired drift, they use keyframes: Every new frame is at first matched to the latest keyframe and as long as there is not too much difference, i.e. the camera has not moved to far, no drift is accumulated. Furthermore, when revisiting a previously seen region old keyframes can enforce additional constraints on the pose graph, also known as loop closures. Although this system is capable of real-time performance on a CPU, it can not do so at a full resolution of 640×480.

ORB SLAM 2 is a state-of-the-art feature-based SLAM system for monocular, stereo and RGBD cameras that runs in real-time on a single CPU. It estimates the camera motion by minimizing the reprojection error and implements loop closure, relocalization, map reuse and bundle adjustment for pose refinement. All these methods show promising results regarding runtime and tracking accuracy, but either generate no [8] or only a sparse global 3D model [9], [4].

One of the first systems to achieve a dense 3D reconstruction in real-time, by exploiting the massively parallel processors on the GPU, was the KinectFusion [10]. The KinectFusion system estimates the current sensor pose with an ICP algorithm and integrates the aquired data via truncated signed distance function (TSDF). In order to achieve real-time performance, the algorithms for both tracking and mapping are fully parallelized. However, the KinectFusion system, which spawned several re-implementations and further works, lacks the scalability for larger scenes due to memory issues (addressing and/or lack thereof).

In order to tackle the problem of a large memory footprint, research on sparse volumetric representations [11], [12] has sprouted. These works successfully use either octrees or hash tables to refer to allocated memory blocks efficiently. One of the works that is able to achieve a very high framerate while reconstructing a dense 3D model is InfiniTAM [7]. It models the 3D world as voxel blocks using a TSDF representation. In order to reduce memory usage, only the scene parts inside the truncation band, i.e. the voxels close to a surface, are usually represented densely in an $8 \times 8 \times 8$ block. A hash table manages these voxel blocks to guarantee a constant lookup time (in case of no collisions).

ElasticFusion [14] reconstructs the 3D world as a number of circular surfels that correspond to the surfaces in the real world. Managing and processing these surfels requires a strong graphics card and is not capable of large-scale reconstructions. Dai et al. proposed BundleFusion [3] that models the world with a voxel block hashing framework [11]. They always optimize over all previously seen frames, generating a globally consistent model. To handle this large amount of data, they utilize two strong graphics cards.

In contrast to [14], [3], we present a framework that generates a globally consistent, dense model in real-time on a consumer graphics card. While pose estimation runs on CPU, the 3D model is generated and processed on the GPU. Most volumetric fusion frameworks do not allow to correct the model [7], [11], [10], e.g. after loop closure (see Fig. 1). We propose several extensions to [7] that enable on-the-fly corrections to generate a globally consistent model in real-time.

III. GLOBALLY CONSISTENT DENSE 3D RECONSTRUCTION

In this paper, we present a real-time 3D reconstruction framework that works with RGBD data. We combine the accurate trajectory estimation of ORB SLAM2 [9] with the volumetric fusion implementation from InfiniTAM v2 [7]. We add novel techniques to InfiniTAM in order to support global model updates that arise from e.g. loop closure. This includes a de-integration method, a global model update step and fusion into the depth maps of keyframes to enable fast real-time processing.

A. Camera Model

We receive an RGBD frame at each time step t that consists of an RGB image I_t and a depth map D_t. We expect I_t and D_t to be aligned and synchronized, such that at a certain pixel position $\vec{x} = (x,y)^\top$ the RGB values are given as $I_t(\vec{x})$ and the corresponding depth as $D_t(\vec{x})$. Then the homogeneous 3D point $\vec{X} = (X,Y,Z,1)^\top$ in the respective camera coordinate system can be computed from $\vec{x}$ and the corresponding depth $Z = D_t(\vec{x})$ with the inverse projection π^{-1}:

$$\pi^{-1}(\vec{x}) = \vec{X} = \left(\frac{x-c_x}{f_x}Z, \frac{y-c_y}{f_y}Z, Z, 1\right)^\top, \tag{1}$$

where f_x, f_y are the focal lengths and c_x, c_y are the principal points. Similarly, the pixel position $\vec{x}$ can be recovered from $\vec{X}$ as:

$$\pi(\vec{X}) = \vec{x} = \left(\frac{X \cdot f_x}{Z} + c_x, \frac{Y \cdot f_y}{Z} + c_y\right)^\top. \tag{2}$$

B. Rigid Body Motion and Warping Function

We restrict the motion between frames to a rigid body motion $g \in SE(3)$. A common representation for g is as

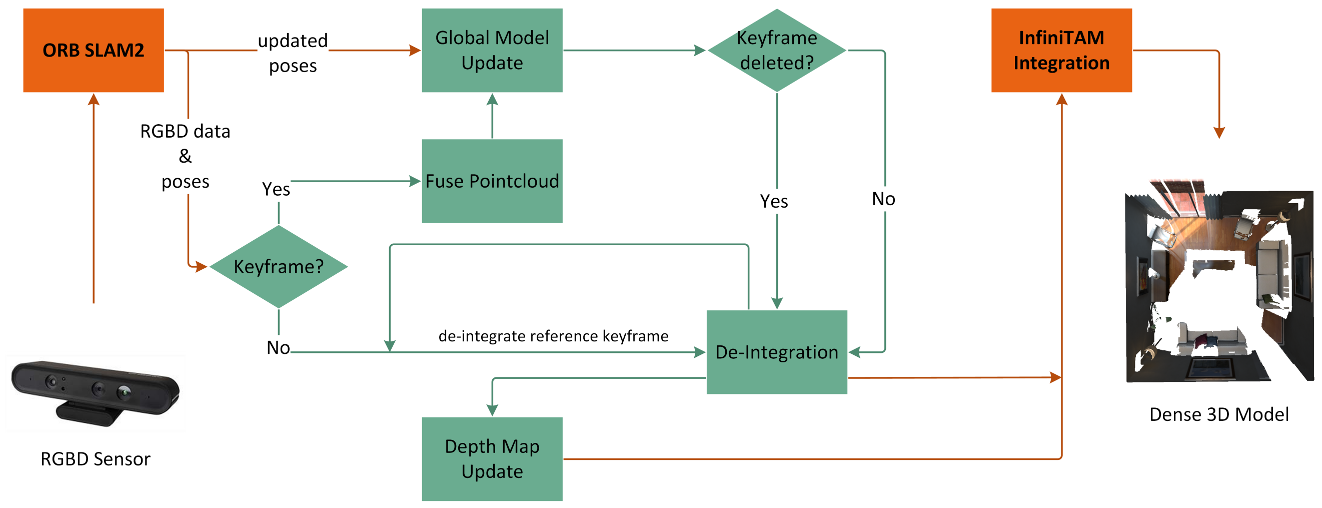


Fig. 2: Our system takes as input the estimated poses from ORB SLAM2 and the RGBD data from the sensor. If the current frame is not a keyframe, we update the corresponding depth map. Otherwise we fuse its depth image with the pointcloud and update the global model. In case that a keyframe has been deleted, we de-integrate it and fuse its information into the next best (closest) keyframe. If not, we de-integrate the frame with its old pose and re-integrate it with its new pose.

transformation matrix T comprising a 3×3 rotation matrix $R \in SO(3)$ and a 3×1 translation vector t:

$$T_{4\times4} = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0 & 1 \end{bmatrix}. \tag{3}$$

The transformation of a point $\vec{X}$ under motion g can be written as:

$$g(\vec{X}) = \vec{X}' = T_{4\times4}\vec{X}. \tag{4}$$

The rigid motion g only has 6 degrees of freedom, thus T with its 12 parameters is over-parametrized. We use a minimal representation as twist coordinates ξ defined by the Lie algebra $se(3)$ associated with the group $SE(3)$. From the 6-vector ξ the transformation matrix T can be recovered by the matrix exponential $T = exp(\xi)$.

We define the full warping function τ that re-projects $\vec{x}$ from frame j with depth $D_j(\vec{x})$ to frame i under the transformation matrix T_{ij} as:

$$\vec{x}' = \tau(\xi_{ij}, \vec{x}, D_j(\vec{x})) = \pi(T_{ij}\pi^{-1}(\vec{x}, D_j(\vec{x}))). \tag{5}$$

C. Combining ORB SLAM2 and InfiniTAM

Fig. 2 shows the interaction between ORB SLAM2 and InfiniTAM. Note that our contributions are depicted in green. We feed the RGBD data (either acquired live via sensor, or from a dataset) into the ORB SLAM2 system and receive the estimated poses for every frame. Then we update our depth maps and adjust the global model if necessary before we integrate the new information into the volumetric representation of InfiniTAM. We explain our depth map and global model updates in detail in the following sections.

D. Model Updates by Re-Integration

We reconstruct our scene geometry by sequentially fusing RGBD data into the TSDF representation. The TSDF is defined as:

$$T(\vec{V}) = max\left(-1, min\left(1, \frac{D_t(\pi(\vec{V})) - Z}{\mu}\right)\right) \quad , \tag{6}$$

where $\vec{V} = (X,Y,Z)^\top$ is a voxel given by its center coordinates, $\pi(\vec{V})$ computes the projection of the voxel onto the depth image, while $D_t(\pi(\vec{V}))$ is the measured depth at the calculated image location. Since $\pi(\vec{V})$ maps the voxel into the camera frame, Z is the distance between the camera and voxel along the optical axis. There are only values between -1 and 1 allowed, corresponding to the distances $-\mu$ and μ respectively (thus truncated). As a result, positive values are assigned to voxels that reside in free space: The closer the voxel is to the surface, the smaller its value. If the voxel lies directly on the surface, its value is set to zero and behind the surface, increasing negative values are assigned.

We adapt the strategy presented by Curless and Levoy [2] and update the TSDF for every new observation i as:

$$\begin{aligned} F_i(\vec{V}) &= \frac{W_{i-1}(\vec{V})F_{i-1}(\vec{V}) + w_i(\vec{V})T(\vec{V})}{W_{i-1}(\vec{V}) + w_i(\vec{V})} \quad , \\ W_i(\vec{V}) &= W_{i-1}(\vec{V}) + w_i(\vec{V}) \quad , \end{aligned} \tag{7}$$

where $W_0(\vec{V}) = 0$ and the uncertainty weight $w_i(\vec{V})$ is usually set to 1, which results in an averaging of the measured TSDF observations.

For a globally consistent reconstruction we need to be able to alter our model with updated camera poses, e.g. when a loop closure is detected. In order to do so, we need to delete old information from the 3D model. We can de-integrate an observation by reversing the operation of (7):

$$\begin{aligned} F_i(\vec{V}) &= \frac{W_{i-1}(\vec{V})F_{i-1}(\vec{V}) - w_i(\vec{V})T(\vec{V})}{W_{i-1}(\vec{V}) - w_i(\vec{V})} \quad , \\ W_i(\vec{V}) &= W_{i-1}(\vec{V}) - w_i(\vec{V}) \quad . \end{aligned} \tag{8}$$

The operations of integrating and de-integrating are symmetric, i.e. one inverts the other. Thus, an observation, if it becomes invalid or updated, can be deleted by de-integrating it from its original pose and re-integrating it with a new pose if necessary.

E. Depth Map Fusion in Keyframes

The idea behind fusing the depth maps of frames into their reference keyframe is to create a system that is able to adapt to global changes within real-time. Without the depth map fusion each frame would have to be re-integrated separately when a model update is induced, while our technique re-integrates only the fused depth maps of keyframes. Since on average only every 10th frame is selected as keyframe, this reduces the amount of operations by a factor of 10. ORB-SLAM2 inserts a keyframe if all of the following conditions are met: (i) more than 20 frames have passed sinice the last global relocalization, (ii) more than 20 frames have passed since the last keyframe insertion or local mapping is idle, (iii) at least 50 keypoints are tracked in the current frame, (iv) more than 10% of keypoints in the current frame are not seen by its reference keyframe. Additionally (for RGBD data), a keyframe is added whenever the number of close keypoints drops below a certain threshold $\tau_t = 100$ and the frame could at least create $\tau_c = 70$ new close keypoints. Fig. 2 depicts our process flow: If a frame is not chosen as keyframe by ORB SLAM2, we fuse its depth map D_c into the depth map of its reference keyframe D_{KF} (see Fig. 3). This update step is closely related to the volumetric fusion integration step presented in (7) :

$$\begin{aligned} D_{KF,i}(\vec{x}') &= \frac{W_{i-1}(\vec{x}')D_{KF,i-1}(\vec{x}') + w_i(\vec{x})Z'}{W_{i-1}(\vec{x}') + w_i} \quad , \\ W_i(\vec{x}') &= W_{i-1}(\vec{x}') + w_i(\vec{x}) \quad , \end{aligned} \tag{9}$$

where $\vec{x}' = \tau(\xi_{ij}, \vec{x}, D_j(\vec{x}))$ is the reprojected pixel position, $Z' = [T_{KF,c}\pi^{-1}(\vec{x}, D_c\vec{x})]_z$ is the z-coordinate of the transformed point, D_c the depth map of the current frame, $T_{KF,c}$ the transformation from current frame to keyframe and the weight $w_i(\vec{x})$ is set equal to 1, which leads to an averaging of the depth values. Please note that we truncate $\vec{x}'$ to always work on integer pixel positions. The difference to the volumetric fusion (7) step is that we update the depth map of the keyframe $D_{KF,i}$ instead of the TSDF values in the model.

In order to not lose any information, we store unfused points in a pointcloud. The pointcloud is represented as a vector, where each entry corresponds to a 3D point, which is transformed into the keyframe coordinates but could not be added to the depth map. Points are not fused into the depth map and added to the pointcloud when either of two conditions arise: (i) The point is transformed out of boundaries variables, i.e. the x and/or y coordinate are negative or larger than the image size or (ii) the depth difference is too large, which can be described as:

$$\left| \frac{1}{D(\vec{x}')} - \frac{1}{Z'} \right| < \Theta_\tau \quad , \tag{10}$$

where Θ_τ is a threshold. This is especially needed on edges in the scene, where it might occur that a point far behind the edge in the new frame would transform onto the edge in the keyframe, e.g. due to rounding. We choose to not update the RGB data which might yield better coloring results but would also increase runtime. Furthermore, unlike depth where invalid measurements can occur, color information is available for every pixel and it is therefore sufficient to color the whole 3D model by just using the RGB image of the keyframe.

Since after every new frame the 3D world model is updated, we need to de-integrated the depth map with the reference keyframe first, then update it with the new depth map of the frame and finally re-integrate it. On the other side, if the current frame is a keyframe, we try to fuse the pointclouds into the new keyframes depth map. In this case every homogeneous 3D point $\tilde{\vec{X}}$ of the pointcloud is transformed into the current keyframe by using the transformation matrix T_{cn} (transforming a point of frame n into the current frame c):

$$\tilde{\vec{X}}' = T_{cn} \cdot \tilde{\vec{X}} \quad , \quad T_{cn} = T_{cw} \cdot T_{wn} \quad , \tag{11}$$

where T_{cw} is the transformation matrix from world coordinates into the current keyframe, and $T_{wn} = T_{nw}^{-1}$ the inverse of the transformation matrix from world coordinates into the keyframe n. We now apply the mapping $\pi(X')$ (2) to get the 2D image coordinates of the current keyframe the 3D point maps to. Finally we can again calculate the update step (9) if the mapped point lies within the image boundaries and satisfies (10). Every point we are able to map in this manner is removed from its pointcloud and if the number of points within a pointcloud falls below a certain threshold Θ_{pc} we delete the whole pointcloud. After this process, we use the depth map as a complemented and smoothed depth image (see Fig. 3), which we integrate into the InfiniTAM model.

F. Global Model Update

The ORB SLAM2 system continuously refines the estimated poses and whenever a new keyframe is selected, we verify the integrated poses from the model with the updated poses. If a significant change occurs, we update our 3D model in real-time. We achieve this model update by de-integrating the depth map with the old pose from the model and re-integrating it with the new pose. In cases, where ORB SLAM2 deletes a keyframe KF_{delete}, we search for the closest keyframe $KF_{closest}$ and de-integrate both from the 3D model. Then we fuse KF_{delete} into $KF_{closest}$ with (9) and re-integrate $KF_{closest}$ into the model.

G. Implementation

Since this process of constant de-integrating and re-integrating can be computationally intensive, we parallelized the update step via CUDA specific code. Note that in a few cases we run into the problem of collision (two or more points in the frame correspond to the same coordinates in the keyframe). In this case, only 1 point will be integrated and the other points are lost. However, this loss of information

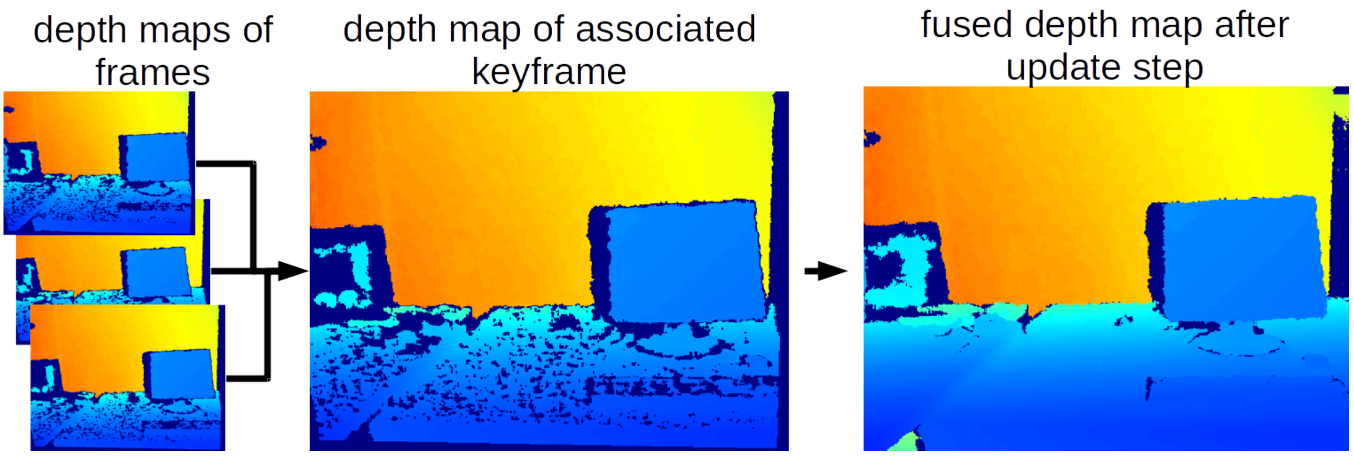

depth maps of frames
depth map of associated keyframe
fused depth map after update step

Fig. 3: Our depth map update complements and smooths the depth map of the keyframe.

can be tolerated for the sake of speed (and not needing any atomic operations). Furthermore, we introduce a ”fast mode”, where frames will only be integrated into the model if a new keyframe is processed, i.e. new frames will only update the depth map of their keyframe but are not directly integrated into the 3D model. A downside to this is that visual feedback provided to the viewer is not immediate but always one keyframe behind. In this manner our system is able to process an average of 15-20 frames per second, where the largest limiting factor is the tracking of ORB-SLAM2 running exclusively on the CPU.

IV. RESULTS AND DISCUSSION

To demonstrate our capabilities we test our system on several real-world image sequences from the TUM RGBD dataset [13] and on the synthetic ICL-NUIM dataset [6]. We evaluate standard InfiniTAM (ITM) [7], ICPCUDA [14], DVO SLAM [8], RGBD SLAM [4] and our method based on ORB SLAM 2 [9]. ICPCUDA is a very fast implementation of ICP with online available code [1]. We run all systems in their standard settings from using the code available online at maximum resolution of 640×480. For RGBD SLAM, we set the feature detector and descriptor type to ORB and extract a maximum of 600 keypoints per frame. In ORB SLAM2, we extract 1000 features per frame with a minimum of 7 per cell and 8 scale pyramid levels. Finally, we run DVO SLAM with its standard 3 scale pyramid levels. We test all the systems on an Intel Core 2 Quad CPU Q9550 desktop computer with 8GB RAM and an NVIDIA GeForce GTX 480. For all models we chose a voxel size of $2cm$ and a truncation band μ of $8cm$ and limited the depth measurements from $0.2m$ to $5.0m$. We empirically found the parameters $\Theta_\tau = 0.005$ and $\Theta_{pc} = 1000$. A high value choice of Θ_τ can result in depth inconsistencies at edges (as stated in III-E), while $\Theta_\tau = 0$ would reject any depth map update. The purpose of Θ_{pc} is to save memory by deleting the whole point cloud if it falls below this threshold. Therefore, a high threshold leads to a deletion of more pointcloud entries and consequently trades a loss of information for memory capacity.

A. Trajectory and Drift Estimation

TABLE I: ATE RMSE on the TUM RGB-D dataset and the synthetic ICL-NUIM dataset *[m]*

	ITM	**ICP CUDA**	**DVO SLAM**	**RGBD SLAM**	**ORB SLAM2**
fr1/desk	0.291	0.144	0.169	0.027	**0.022**
fr1/desk2	0.483	0.273	0.148	0.041	**0.023**
fr1/room	0.523	0.484	0.219	0.104	**0.069**
fr1/xyz	0.032	0.042	0.031	0.017	**0.010**
fr2/desk	0.114	1.575	0.125	0.092	**0.079**
fr2/xyz	0.042	0.223	0.021	0.016	**0.013**
fr3/office	1.258	1.161	0.120	0.034	**0.011**
fr3/nstn	1.979	1.666	0.039	0.051	**0.018**
lr/kt0	0.045	0.697	**0.006**	0.011	0.008
lr/kt1	0.009	0.045	**0.005**	0.013	0.162
of/kt0	0.054	0.205	**0.007**	0.029	0.027
of/kt1	0.025	0.275	**0.004**	0.724	0.051

We use the evaluation tools provided by [13] to calculate the absolute trajectory error (ATE) and the relative pose error (RPE). As suggested in [13], we compare the root mean squared error (RMSE) of the ATE and RPE. The ATE directly compares the absolute distances of the trajectory in the ground truth file and the output trajectory of the various systems. This is a good measurement for global consistency in SLAM systems. Let $P_{1:n}$ be the estimated trajectory and $Q_{1:n}$ the ground truth trajectory. Then we can find a least-squares solution for the rigid-body transformation S which maps $P_{1:n}$ onto $Q_{1:n}$ and compute the absolute trajectory error at time step i:

$$F_i := Q_i^{-1} S P_i \quad . \tag{12}$$

Table I shows the results for the ATE RMSE where ORB-SLAM2 outperforms all other systems on the TUM RGBD sequences. On the ICL-NUIM datasets, DVO-SLAM outshines ORB-SLAM2. This is due to the synthetic nature of the datasets, where perfect depth values allow a very accurate tracking for DVO-SLAM, whilst ORB-SLAM2 still needs to rely on the extracted ORB features. The high error value

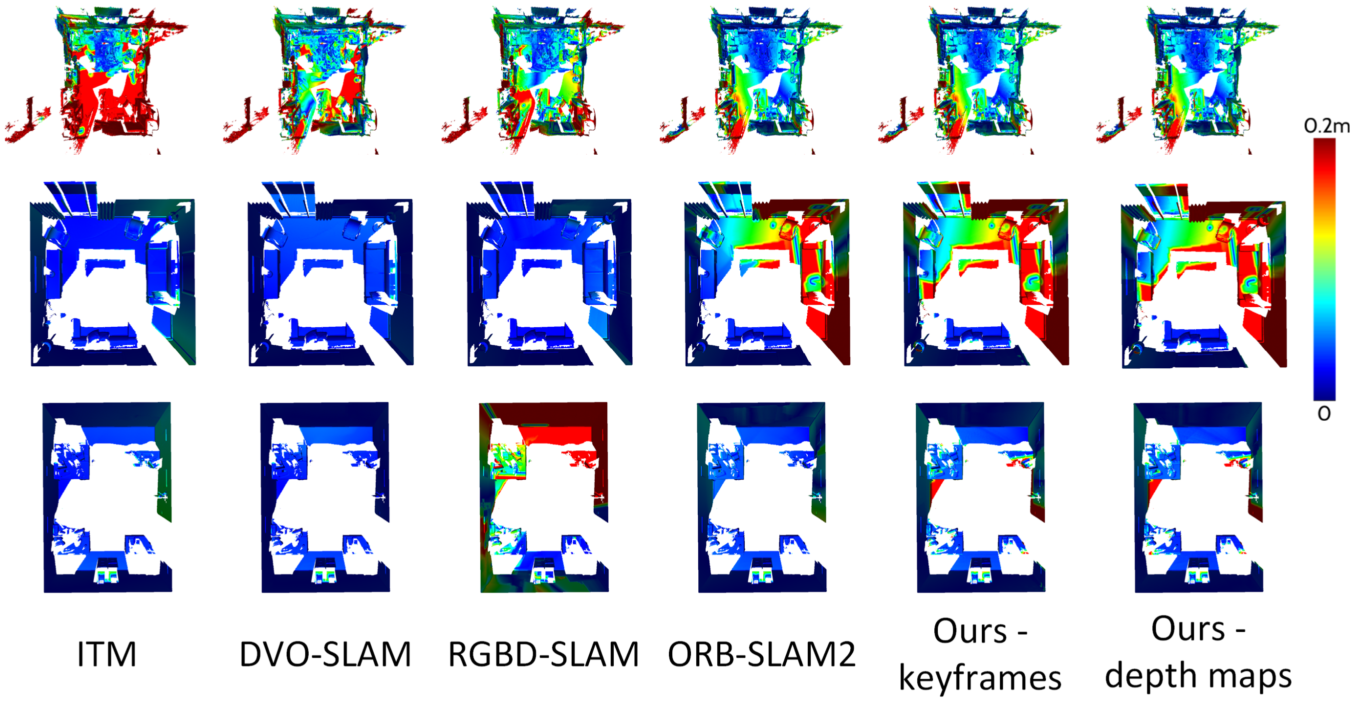
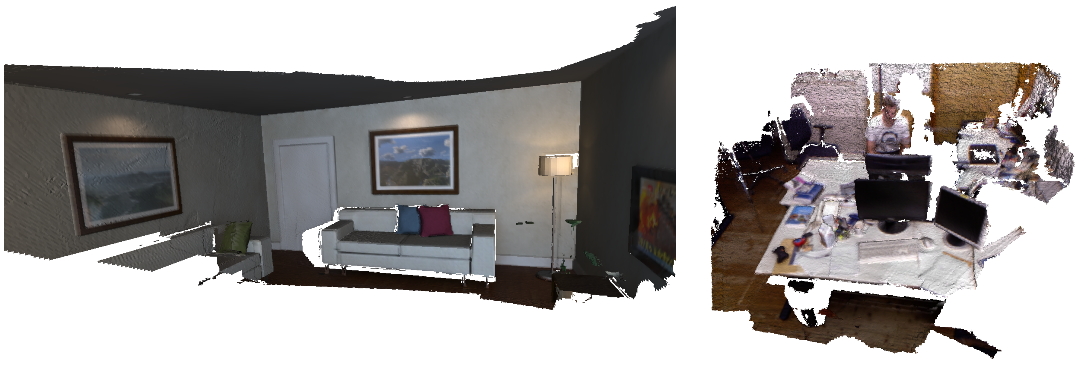


Fig. 4: Surface Reconstruction: Heat maps depicting the error from the ground truth model to the estimated model. Datasets from top to bottom: fr1/room, lr/kt1, of/kt1.



(a) lr/kt0 (b) fr1/xyz

Fig. 5: Sample reconstruction models of our approach from the TUM RGBD and ICL-NUIM datasets.

for the lr/kt1 sequence with ORB-SLAM2 is a result of not revisiting any structure and therefore being unable to perform a loop closure. The RPE computes the relative difference of the trajectory over a fixed time interval Δ. In visual odometry systems it evaluates the drift between frames and in SLAM systems it can measure the accuracy at loop closures. The RPE at time step i is defined as:

$$E_i := (Q_i^{-1}Q_{i+\Delta})^{-1}(P_i^{-1}P_{i+\Delta}) \quad . \tag{13}$$

We choose to evaluate the RPE in Table II over the time interval of 1 second ($\Delta = 1s$). Here again, the synthetic ICL-NUIM datasets show slightly better results for the other systems compared to ORB-SLAM2. In order to be able to use the TUM tools, we converted all datasets into the TUM format, i.e. we changed the image and ground truth formats and added the associate files which can also be generated with the provided tools. In cases where the algorithm is non-deterministic, i.e. the result trajectories differ for every run, we execute the algorithm 10 times and take the mean value. Algorithms which belong to this category are ORB SLAM2 and RGBD SLAM.

B. *Surface Reconstruction Accuracy*

To measure the surface reconstruction accuracy, we calculate the one-sided Hausdorff distance from the groundtruth

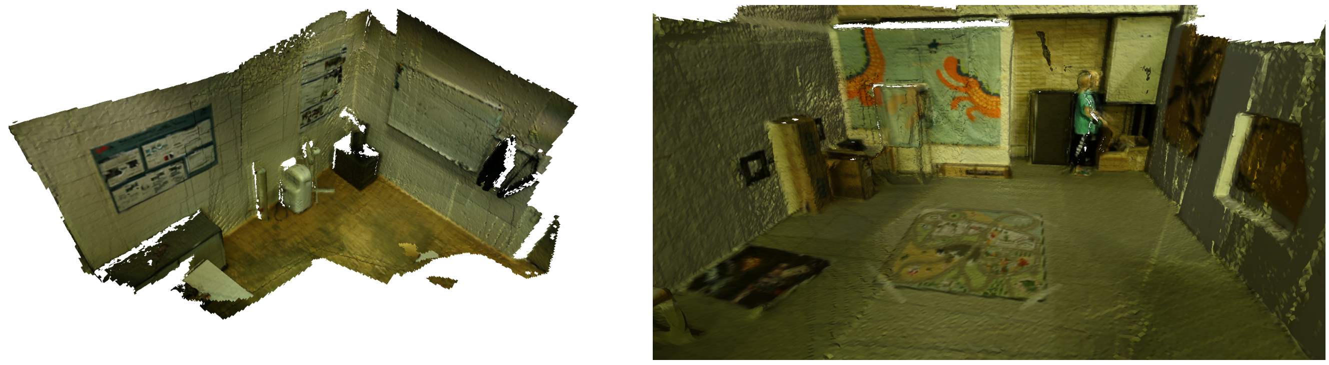
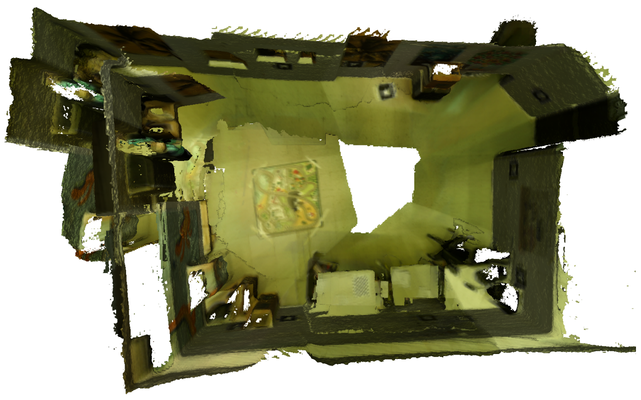


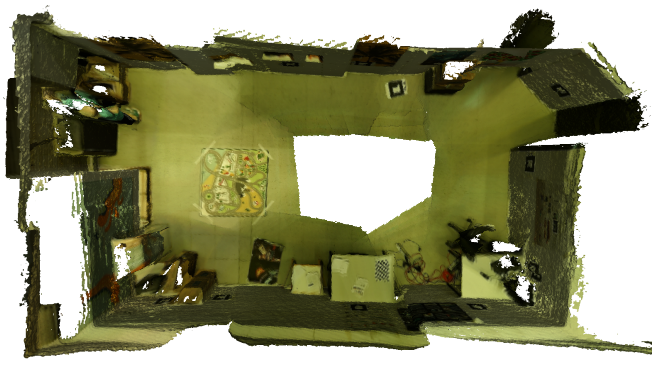
Fig. 6: Sample reconstruction models of our approach from our own Orbbec Astra Pro recordings.

TABLE II: Translational RPE RMSE on the TUM RGB-D dataset and the synthetic ICL-NUIM dataset with $\Delta = 1s$ $[\frac{m}{s}]$

	ITM	ICP CUDA	DVO SLAM	RGBD SLAM	ORB SLAM2
fr1/desk	0.207	0.100	0.052	0.036	**0.026**
fr1/desk2	0.327	0.164	0.061	0.045	**0.033**
fr1/room	0.259	0.129	0.056	0.053	**0.048**
fr1/xyz	0.047	0.031	0.024	0.027	**0.016**
fr2/desk	0.024	0.109	0.016	0.018	**0.012**
fr2/xyz	0.007	0.027	0.005	0.006	**0.004**
fr3/office	0.052	0.131	0.017	0.016	**0.009**
fr3/nstn	0.242	0.263	0.017	0.019	**0.015**
lr/kt0	0.005	0.140	**0.002**	0.003	0.008
lr/kt1	**0.001**	0.017	0.002	0.002	0.074
of/kt0	**0.003**	0.061	**0.003**	0.005	0.016
of/kt1	**0.002**	0.152	**0.002**	0.007	0.034



(a) Original InfiniTAM



(b) Our approach

Fig. 7: Sample reconstruction of a room recorded and reconstructed in real-time with our Orbbec Astra Pro. (a) shows the original InfiniTAM reconstruction, which is unable to adapt the model to loop closure (see top left corner). (b) depicts our approach with a globally consistent model.

TABLE III: Evaluation of the surface reconstruction accuracy: Hausdorff distances from the ground truth surface to the reconstructed surfaces *(m)*.

	ITM	DVO SLAM	RGBD SLAM	Ours		
				all frames	keyframes	depth maps
fr1/desk	0.067	0.071	0.037	**0.033**	0.037	0.034
fr1/desk2	0.091	0.088	0.078	**0.043**	0.051	0.048
fr1/room	0.228	0.152	0.164	**0.084**	0.091	0.087
fr1/xyz	0.033	0.046	0.019	**0.012**	0.017	0.015
lr/kt0	**0.004**	0.005	0.006	0.008	0.016	0.016
lr/kt1	0.015	**0.007**	0.008	0.097	0.114	0.113
of/kt1	0.014	**0.006**	0.095	0.017	0.027	0.025

3D model to the reconstructed 3D model:

$$d_H(X,Y) = \sup_{x\in X}\inf_{y\in Y} \mathrm{d}(x,y) \quad , \qquad (14)$$

where X is the set of groundtruth vertices, Y the set of the reconstructed vertices and $\mathrm{d}(x,y)$ is the Euclidian distance between the two vertices x and y. We sample each vertex in X, find the distance to the closest point in Y and take the average. Table III lists the result of this process for different datasets and methods. ORB SLAM2 outperforms all other systems on the freiburg1 datasets when integrating the model frame by frame without using de-integration (all frames). However, note that we used already optimized trajectories for this test and thus no pose updates had to be incorporated. When we only integrate keyframes into the model, i.e. all non keyframes will not be processed by the system, the reconstruction error increases slightly. We counter this effect by using our fused depth maps. On the ICL-NUIM datasets, InfiniTAM and DVO SLAM outshine ORB SLAM2. This is due to the synthetic nature of the datasets, where perfect depth values allow a very accurate tracking for the former two, whilst ORB SLAM2 still needs to rely on the extracted ORB features. Furthermore, we can see in Fig. 4 that no loop closure could be performed in the lr/kt1 dataset (due to not revisiting any structure), which leads to a larger error. Note that in the of/kt1 dataset our method shows some areas with an increased error. The reason for this is that no keyframe was detected there and consequently no values exist.

For further qualitative evaluation we tested our system on several well known datasets (see Fig. 5) and also on datasets recorded with our own Orbbec Astra Pro (see Fig. 6). The whole extend of our method is illustrated in Figure 7: The original InfiniTAM is unable to adapt the model to global updates and therefore structures can appear at the wrong places, e.g. the reconstruction of 2 walls on the left and the tables at the bottom. With our approach we obtain a globally consistent model.

V. CONCLUSIONS

In this paper we presented a real-time capable method to combine the tracking accuracy of a state-of-the-art SLAM system [9] with the dense model generation of a volumetric fusion system [7]. We utilize the depth maps of all frames but fuse them into the depth map of their corresponding keyframes. The fused depth map is then integrated into the 3D model instead of every single frame, resulting in a speedup of about a factor of 10. Using fewer keyframes can increase the speedup even further, but will also impact the quality of the model, especially if translation and rotation between keyframes becomes very large. In this manner our system is able to adapt the model online, when updated poses are available, e.g. after loop closure or bundle adjustment. For real world data we have shown that our method yields excelling results, especially when compared to the original InfiniTAM ICP approach. Note that our system is not limited to ORB SLAM2, but can in theory work with any keyframe based tracking method. Therefore, it could enable means for a globally consistent dense real-time 3D reconstruction for many different SLAM and VO systems often lacking this feature.

REFERENCES

[1] "Icpcuda," https://github.com/mp3guy/ICPCUDA, 2018, [Accessed 20-March-2018].

[2] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1996, pp. 303–312.

[3] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, p. 24, 2017.

[4] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[5] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[6] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[7] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile devices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 11, pp. 1241–1250, 2015.

[8] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 2100–2106.

[9] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[10] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2011, pp. 127–136.

[11] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 6, p. 169, 2013.

[12] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3d mapping in real-time on a cpu," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2021–2028.

[13] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 573–580.

[14] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 14, pp. 1697–1716, 2016.