

CodeMapping: Real-Time Dense Mapping for Sparse SLAM using Compact Scene Representations

Hidehobu Matsuki, Raluca Scona, Jan Czarnowski and Andrew J. Davison

Abstract—We propose a novel dense mapping framework for sparse visual SLAM systems which leverages a compact scene representation. State-of-the-art sparse visual SLAM systems provide accurate and reliable estimates of the camera trajectory and locations of landmarks. While these sparse maps are useful for localization, they cannot be used for other tasks such as obstacle avoidance or scene understanding. In this paper we propose a dense mapping framework to complement sparse visual SLAM systems which takes as input the camera poses, keyframes and sparse points produced by the SLAM system and predicts a dense depth image for every keyframe. We build on CodeSLAM [1] and use a variational autoencoder (VAE) which is conditioned on intensity, sparse depth and reprojection error images from sparse SLAM to predict an uncertainty-aware dense depth map. The use of a VAE then enables us to refine the dense depth images through multi-view optimization which improves the consistency of overlapping frames. Our mapper runs in a separate thread in parallel to the SLAM system in a loosely coupled manner. This flexible design allows for integration with arbitrary metric sparse SLAM systems without delaying the main SLAM process. Our dense mapper can be used not only for local mapping but also globally consistent dense 3D reconstruction through TSDF fusion. We demonstrate our system running with ORB-SLAM3 and show accurate dense depth estimation which could enable applications such as robotics and augmented reality.

Index Terms—SLAM, Mapping, Vision-Based Navigation

I. INTRODUCTION

Vision based Simultaneous Localization and Mapping (SLAM) has made remarkable progress over the past 20 years and has a wide range of commercial applications such as robot navigation and AR/VR. Prior research has shown that the key elements for SLAM accuracy are informative point selection (typically corners or edges) and joint optimization of the camera poses and landmark locations (bundle adjustment). Thanks to these, sparse SLAM produces more accurate results compared to dense methods which decouple map building and camera pose estimation [2][3][4]. When used in practical applications, sparse SLAM is often fused with other sensors to improve robustness and also provide metric scale [4] [5]. However, the sparse feature map is not visually informative

Manuscript received: February, 24, 2021; Revised May, 29, 2021; Accepted June, 28, 2021.

This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments.

Research presented in this paper has been supported by Dyson Technology Ltd.

The authors are with Dyson Robotics Laboratory, Imperial College London, United Kingdom h.matsuki20@imperial.ac.uk

Digital Object Identifier (DOI): see top of this page.

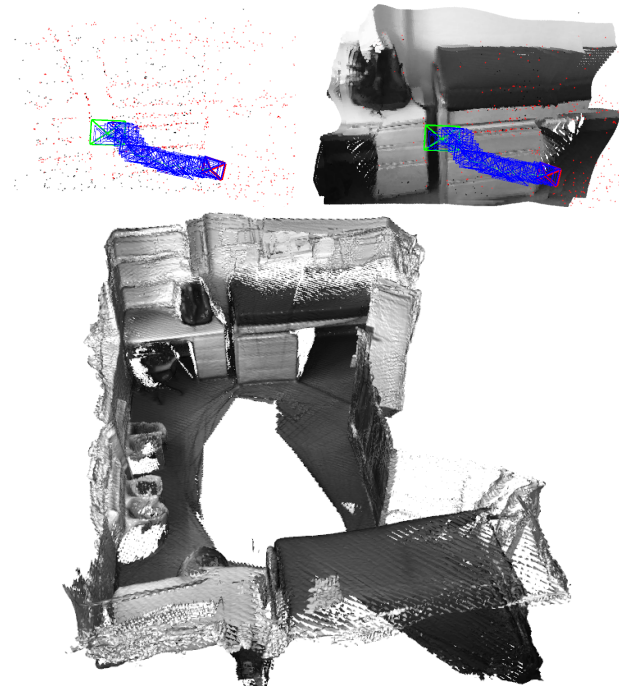


Fig. 1. **Top:** Real-time dense mapping result (*left:* sparse SLAM, *right:* ours). Our dense mapper completes the sparse scene geometry and refines it by sliding-window multi-view optimization **Bottom:** An example of global 3D reconstruction by TSDF fusion of our mapping system. Our method can be applied to arbitrary metric sparse SLAM systems.

and cannot directly be used for tasks such as collision-free motion planning or surface-aware AR. Several methods have attempted to perform dense mapping and camera localization simultaneously [6][7], but they were not as successful as the sparse methods because (1) dense image alignment is vulnerable to photometric noise and (2) the large number of scene parameters restricts real-time joint optimization. Therefore, instead of running dense SLAM, it can be more practical to use a separate algorithm to infer dense depth. This densification has commonly been done by Multi-View Stereo algorithms but in recent years many depth completion methods using deep learning have also been investigated and show promising performance [8][9][10].

In this paper we propose *CodeMapping*, a novel real-time dense mapping method which leverages the geometric prior information provided by sparse SLAM. We extend the compact scene representation proposed in CodeSLAM [1] to complete sparse point sets. We train a VAE to predict a dense depth map and condition it on the corresponding gray-scale image, a sparse depth image from the observable features in the map

and a reprojection error image, to enable uncertainty-aware depth completion. Consecutive depth maps are refined through multi-view optimization using fixed camera poses provided by the SLAM system, to improve their consistency and accuracy. Our system works in parallel to the SLAM system, subscribing to the estimated camera poses, sparse depth and reprojection error maps. This formulation enables flexible keyframe selection for multi-view dense bundle adjustment and real-time execution without interrupting the main SLAM process.

II. RELATED WORK

Visual SLAM: SLAM methods proposed in the early 2000s used a Kalman Filter to estimate the camera pose and the map[11]. However, since filtering-based SLAM includes all landmark positions and camera poses in a state space and updates all the variables at every time step, the method can not be applied to large scale scenes due to computational limitations. A major breakthrough was made in 2007 by PTAM [12], a keyframe-based SLAM method which runs tracking and mapping in separate threads. Similarly, many current SLAM algorithms also use keyframes and apply multi-threading to separate real-time locally consistent tracking and slower globally consistent mapping into different threads. Various types of scene representations have been proposed for keyframe-based SLAM, but most current methods are using sparse 3D points. This is because joint optimization of poses and geometry is necessary for accurate performance and a sparse set of points ensures real-time operation. Dense geometry is desirable for scene understanding but due to the large number of parameters normally used to represent it, joint optimization in real-time has not been possible. CodeSLAM [1] and DeepFactors [13] proposed a dense compact scene representation using VAE, but they predict a dense depth map purely from a gray-scale image which can be inaccurate in practice (Fig. 2).

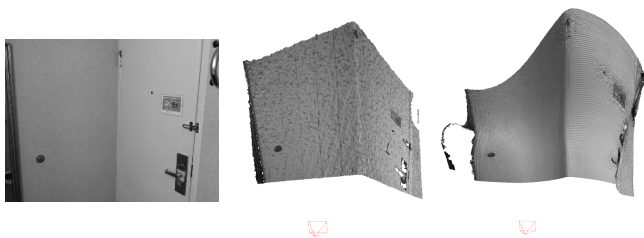


Fig. 2. An example of depth image prediction using DeepFactors [13] (**Left:** Input image, **Middle:** 3D visualization of depth from an RGBD sensor, **Right:** 3D visualization of the depth image estimated with DeepFactors). We can clearly see that DeepFactors’ prediction is incorrectly bent around the wall and the door.

SLAM with Scale Estimation: In practice, Visual SLAM is rarely used without scale information. Metric scale is recovered using additional sensors or priors such as IMUs [4][14][15], wheel odometry [5], known camera height [16], stereo/depth cameras [4][17]. Recently, CNN-based depth predictions are used to infer metric scale for monocular visual SLAM [18][19]. Using scale information, camera trajectory estimates from sparse VO/SLAM systems exhibit errors smaller than a few percent of the total traveled distance. These

systems can be used for a variety of tasks such as mobile robot navigation and position tracking for AR/VR.

Depth Completion using Deep Learning: Reconstruction of dense depth from sparse points was classically formulated as an energy minimization problem [20]. With the recent development of deep learning, learning-based depth completion methods have shown promising performance. For example, Ma *et al.* [8] proposed a color-guided encoder-decoder network and revealed that sparse-point guided depth prediction shows much better performance than purely image-based depth prediction. Zhang *et al.* [9] used predicted surface normals to densify Kinect depth images, and Xiong *et al.* [10] applied a Graph Neural Network to depth completion. Cheng *et al.* [21] proposed a semantic scene completion network for LiDAR point clouds for a large scale outdoor environment. However, these networks are designed for single-view densification and do not actively address multi-view consistency when processing video streams. Recent techniques propose to estimate dense 3D reconstructions from posed images. For example, Atlas [22] directly predicts TSDFs and DeepMVS [23] predicts disparity maps. While both methods produce impressive reconstructions, they require significant amounts of GPU memory to run 3D CNNs in the case of Atlas and plane-sweeping volumes in DeepMVS. Regarding DeepMVS, the large structure of the network also makes this method more suited to offline applications. Our approach is more modular and composed of a number of steps. The different modules can be used in different tasks depending on the run-time and memory requirements. For example, in time-critical tasks such as obstacle avoidance, only the single-view depth completion network may be sufficient. If a dense model is required, the multi-view optimization and TSDF fusion can be used at an increased computational cost.

Real-time Dense Mapping: The work most similar to ours is CodeVIO[24], which uses a CodeSLAM-like VAE conditioned on an intensity image and a sparse depth map. Their system is tightly coupled with filtering based visual-inertial odometry and simultaneously estimates scene geometry and camera trajectory in the same state space. Compared to their system, our method has the following advantages: first, our network is conditioned not only on intensity and depth but also on the reprojection error of points, enabling the network to consider the confidence of SLAM points when estimating dense depth. Second, our system runs as a separate backend process and is loosely-coupled with a keyframe-based SLAM system. Keyframe-based SLAM can estimate a larger number of points in real-time when compared to filtering which is important for sparse-to-dense networks as more information provided to the network simplifies the learning task and can lead to more accurate results. Third, since keyframe-based sparse SLAM provides a globally consistent camera trajectory, our depth maps can be used not only for local mapping but also global depth map fusion.

III. SYSTEM OVERVIEW

Fig. 3 shows an overview of our system. The system launches a SLAM process (tracking, local and global mapping

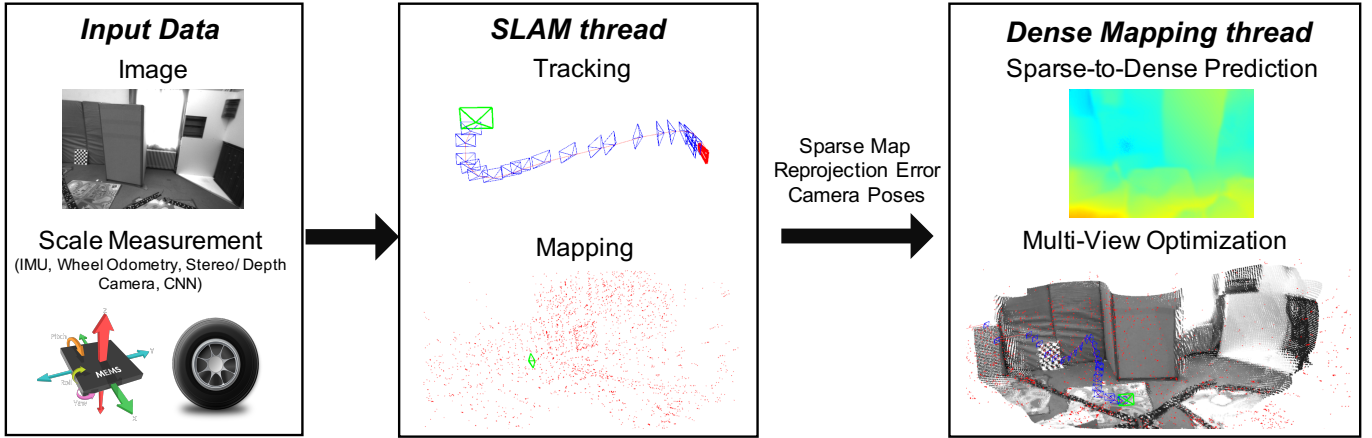


Fig. 3. System overview of CodeMapping: the dense mapper subscribes to bundle-adjusted sparse point sets and camera poses to run dense prediction and multi-view optimization in a separate thread.

threads) and a Dense Mapping process which continuously runs in parallel until the input data stream stops. In this work, we use ORB-SLAM3 [4] as a sparse SLAM system because it supports multimodal sensor input and shows state-of-the-art performance. We describe these components next.

A. SLAM Thread

Tracking: Tracking uses sensor information to estimate the pose of the camera relative to the existing map. ORB-SLAM3 runs pose-only bundle adjustment which minimizes the reprojection error of matched features between the current image and the map. This process is done for every input frame in real-time and it also decides whether the current frame becomes a keyframe to be included in the map.

Mapping: Mapping runs bundle adjustment for pose and geometry refinement. ORB-SLAM3 runs local and global bundle adjustment for this process. Local bundle adjustment uses only a small number of keyframes for optimization, while global bundle adjustment uses the whole graph. After every local bundle adjustment, the SLAM thread transfers data from a window of four keyframes to the Dense Mapping thread. The four keyframes consist of the latest keyframe and the top three covisible keyframes based on the ORB-SLAM3’s covisibility criteria. The keyframe data consists of respective camera poses, sparse depth and reprojection error images. Each sparse depth map is obtained by projecting 3D landmarks on the keyframe and the reprojection error map is calculated by projecting each 3D landmark to all observed frames and computing the average distance to the corresponding matched ORB feature location. The reprojection error of newly registered keypoints with no matches from other frames are initialized with a large value (10).

B. Dense Mapping Thread

Sparse-to-Dense Prediction: When the dense mapping thread receives keyframe data from the SLAM thread, it checks whether the keyframe data has been previously processed. If not, the system runs the depth completion VAE using the the sparse depth and reprojection error input images to predict

an initial dense depth map. The depth completion VAE also generates low dimensional latent codes for the predicted dense depth maps, which are used later in multi-view optimization. This is described in Sec. IV and the runtime of this process is provided in Table III.

Multi-view Optimization: After an initial prediction, the depth maps are refined by sliding window multi-view optimization using the camera poses provided by the SLAM system (Fig. 4). We use the factor-graph based optimization proposed in DeepFactors, which considers different types of factors between overlapping frames. We optimize only the per-frame codes, but not the camera poses as we assume the poses from sparse SLAM are already accurate enough and pixel-wise dense alignment is less likely to improve the accuracy further. We describe this in detail in Sec. V.

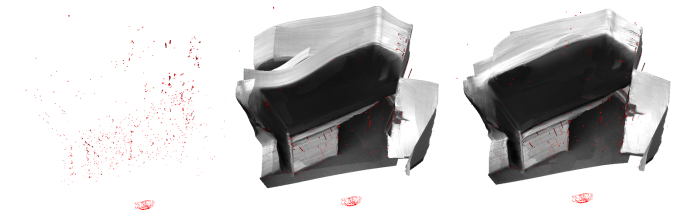


Fig. 4. An example of 5-view depth completion and multi-view optimization (**left:** sparse SLAM points, **middle:** initial depth completion, **right:** 5-view optimization). Multi-view optimization improves depth estimation accuracy especially in texture-less regions such as the white wall.

IV. SPARSE-TO-DENSE PREDICTION

A. Network Architecture

We extend the conditional variational autoencoder from DeepFactors and our network architecture is shown in Fig. 5. The top network is a U-Net [25] that takes as input a gray-scale image concatenated with a sparse depth and reprojection error maps from the sparse SLAM system. Similarly to CodeSLAM and DeepFactors, the depth and reprojection error values are normalized to the range [0, 1] using the proximity parametrization [1]. The network uses the input to compute deep features by applying multiple convolution layers and outputs the depth uncertainty prediction. As Fig. 6 shows,

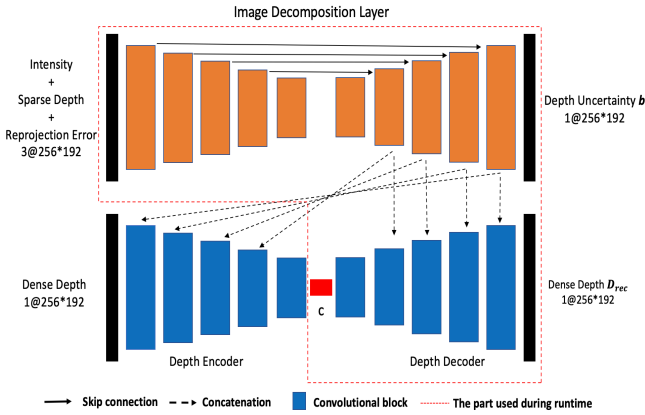


Fig. 5. Architecture of the proposed conditional VAE network.

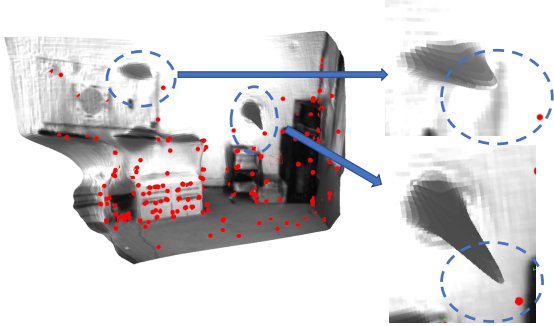


Fig. 6. Depth prediction without reproj error conditioning. Red points are generated by sparse SLAM and the network completes the dense geometry. As highlighted in blue circles, the prediction is severely affected by outlier SLAM points.

sparse SLAM landmarks can contain outliers and trusting all points equally can cause spikes in the predicted depth map. We observe that outlier points tend to have a higher average reproj error value compared to inliers. Hence, we include the sparse reproj error map in the input of the U-Net to provide the network with per-point uncertainty. The bottom network is a VAE conditioned by the decomposed features. The network generates latent code c , dense depth prediction D and uncertainty map b . The loss function of the network consists of a depth reconstruction loss and a KL divergence loss. The depth reconstruction loss is defined as:

$$\sum_{\mathbf{x} \in \Omega} \frac{\|D[\mathbf{x}] - D_{gt}[\mathbf{x}]\|}{b[\mathbf{x}]} + \log(b[\mathbf{x}]), \quad (1)$$

where \mathbf{x} is a 2D pixel coordinate on the image plane Ω and D_{gt} is the ground truth depth image.

B. Training Procedure

We used $\sim 0.4M$ images from the ScanNet dataset [26] following the official training/test split. ScanNet contains a variety of indoor RGBD images taken with a Kinect camera. To generate the sparse depth input, we run ORB keypoint detection on the color image and use these detections to select 1000 keypoints from each Kinect depth map. The input of the VAE is a raw Kinect depth image, and we use the depth map obtained by rendering the given 3D scene model from the current pose as ground truth depth for the loss function. Generating a reproj error map is less trivial as it would

not be feasible to run a SLAM system over the entire ScanNet dataset due to the long processing time. Instead, we propose a method for simulating reproj errors during training. We observe that the distribution of reproj errors from ORB-SLAM3 is similar between different sequences and closely resembles an exponential Gaussian distribution (Fig. 7). We propose that adding noise to the sparse depth map so that resulting reproj errors follow this distribution allows us to accurately simulate the properties of ORB-SLAM3.

The dataset creation steps are detailed in Fig. 8. Firstly, for every frame in the dataset (“reference frame”), we randomly select one neighboring frame which is within 2 meters from the reference (“virtual keyframe”). Then, we unproject points from the reference frame to 3D and compute the line that connects each 3D point to the virtual keyframes’ camera center. For every point, we sample a reproj error to simulate from the exponential Gaussian distribution. We use this to perturb a point’s location along this line so that the distance between the 2D projection of the perturbed point and the 2D location of the original point on the reference frame matches this sampled reproj error value. We apply this process for the all sparse depth measurements and use the perturbed sparse depth map and reproj error as input to the U-Net. We trained the network for 10 epochs with a learning rate of 0.0001, image size of 256×192 , code size 32 and using the Adam Optimizer.

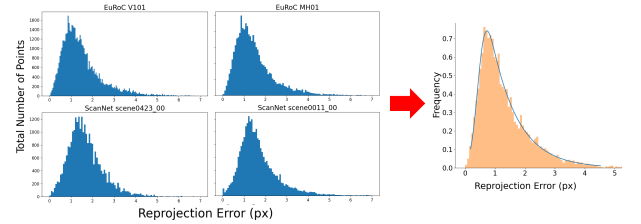


Fig. 7. **Left:** The histogram of ORB-SLAM reproj errors over different sequences. We can observe that the distribution is similar between sequences. **Right:** Probabilistic Model Fitting of the reproj error histogram. We model the distribution by exponential Gaussian distribution ($K=4.31$, $Mean=0.44$, $Sigma=0.20$)

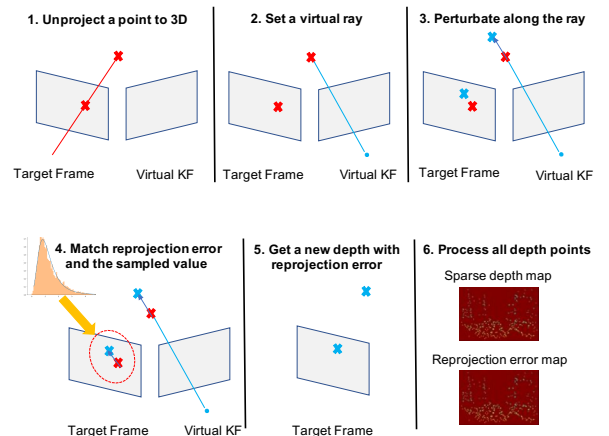


Fig. 8. Procedure to simulate per-point noise following the modeled reproj error distribution described in Fig. 7.

V. MULTI-VIEW OPTIMIZATION

We improve the accuracy and consistency of per-frame depth predictions by running multi-view dense bundle adjustment (Fig. 4). Following the methodology proposed in DeepFactors, we optimize depth images on a compact manifold generated by the conditional VAE. Using the code representation, we minimize a set of different objective functions between consecutive frames that ensure consistency of observations from the overlapping frames and find the best estimate of scene geometry. We formulate this problem as a factor graph (Fig. 9), where the variables are the per-image codes and the factors are the error terms described next. We use DeepFactors’ optimizer implemented using the GTSAM library [27] to solve this problem. The following sections describe these factors in greater detail. We use Huber norm on all factors as a robust cost function.

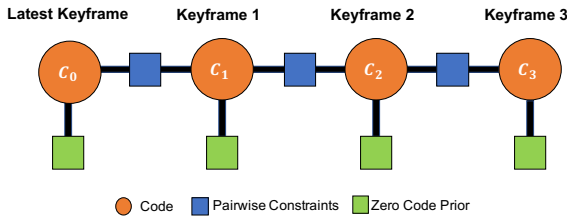


Fig. 9. Factor graph formulation of multi-view optimization. The optimization window consists of 4 keyframes and constraints are added between consecutive pairs of frames. We include a zero-code prior for regularization.

A. Photometric Factor

The Photometric Factor optimises for photometric consistency between images I_i and I_j . It computes the sum of pixel-wise squared differences formulated as:

$$photoE^{ij}(\mathbf{c}_i) = \sum_{\mathbf{x} \in \Omega} \|I_i[\mathbf{x}] - I_j[\omega_{ji}(\mathbf{x}, \mathbf{c}_i, I_i)]\|^2, \quad (2)$$

where \mathbf{x} is a 2D pixel coordinate on the image plane Ω and ω_{ji} is a function which warps this coordinate from frame i to frame j :

$$\omega_{ji}(\mathbf{x}, \mathbf{c}_i, I_i) = \pi(\mathbf{T}_{ji}(\pi^{-1}(\mathbf{x}, D_i[\mathbf{x}]))). \quad (3)$$

The π function projects a 3D point onto a 2D plane while π^{-1} is the inverse operation. $D_i = D(\mathbf{c}_i, I_i, S_i, R_i)$ is the depth map decoded from code $\mathbf{c}_i \in \mathbb{R}^{32}$, intensity image I_i , sparse depth image S_i and sparse reprojection error image R_i . $\mathbf{T}_{ji} \in \text{SE}(3)$ is the 6 DoF transformation from frame i to frame j .

B. Reprojection Factor

The Reprojection Factor minimizes the reprojection error of matched keypoints between frames. Similar to DeepFactors, we use BRISK as a keypoint detector and descriptor. The objective function is defined as:

$$repE^{ij}(\mathbf{c}_i) = \sum_{\mathbf{x}, \mathbf{y} \in M_{ij}} \|\omega_{ji}(\mathbf{x}, \mathbf{c}_i, I_i) - \mathbf{y}\|^2 \quad (4)$$

where M_{ij} is a set of matched feature points between frames i and j . \mathbf{x} and \mathbf{y} are their corresponding 2D image coordinates on frames i and j respectively.

C. Sparse Geometric Factor

The Sparse Geometric Factor directly measures the per-pixel difference between the depth maps after warping them to a common reference frame. Due to computational limitations, we uniformly sample a set of pixels on the image plane for which we compute this error (Ω'). The factor helps to improve the consistency of textureless regions where the photometric and reprojection factors do not provide sufficient signal for optimization. It is formulated as:

$$d_{pt}E^{ij}(\mathbf{c}_i) = \sum_{\mathbf{x} \in \Omega'} \|\mathbf{T}_{ji}\pi^{-1}(\mathbf{x}, D_i[\mathbf{x}])|_z - D_j[\hat{\mathbf{x}}]\|^2, \quad (5)$$

where D_i and D_j are the depth maps decoded for frames i and j as described in Section V-A, $\hat{\mathbf{x}} = \omega_{ji}(\mathbf{x}, \mathbf{c}_i, I_i)$ and $|\mathbf{v}|_z$ denotes taking the z component of the vector \mathbf{v} .

VI. EXPERIMENTAL RESULTS

We trained our model with a ScanNet training split using the Python TensorFlow library. We export the trained model to the C++ SLAM system using the TensorFlow C API. Similarly to DeepFactors, our dense optimization module has been implemented in C++ with CUDA. We run all experiments in an Intel Core i9-10900 CPU at 2.8 GHz and NVIDIA GTX 3080 GPU. We have used an image resolution of 256×192 for dense mapping. We evaluate our system on the ScanNet and EuRoC MAV datasets.

A. Quantitative Depth Accuracy Evaluation

We compared our method against DeepFactors [13] (where depth image predictions are only conditioned on intensity images) and the depth completion method proposed by Ma *et al.* [8] (single-view depth completion given a set of sparse points). We trained the network model from Ma *et al.* [8] on the ScanNet training set with 1000 depth points per image. As an evaluation metric, we measured mean absolute error (MAE) and root mean square error (RMSE) between the predicted and the ground truth depth images. We also evaluate the depth prediction of our method under different configurations, including with and without reprojection error conditioning or multi-view optimisation.

1) *ScanNet Dataset*: We run ORB-SLAM3 in RGBD mode to estimate per-keyframe camera poses, sparse depth and reprojection error maps consisting of 200 to 1000 points per frame. We run our system on the test sequences of this dataset and evaluate the depth prediction accuracy against ground truth depth images from the renderings of the provided 3D models. We report our results in Table I. Our method outperforms the other dense prediction networks in all sequences. Multi-view optimization further improves our single-view dense prediction by approximately 10%. This can be seen in Fig. 10, where we show a comparison of TSDF fusion of depth images estimated with and without multi-view optimization. Optimization leads to more accurate and also more consistent depth maps that produce higher quality 3D reconstructions. On ScanNet, conditioning on reprojection error does not lead to significant performance improvement due to ORB-SLAM3 generally working well and not producing gross outliers.

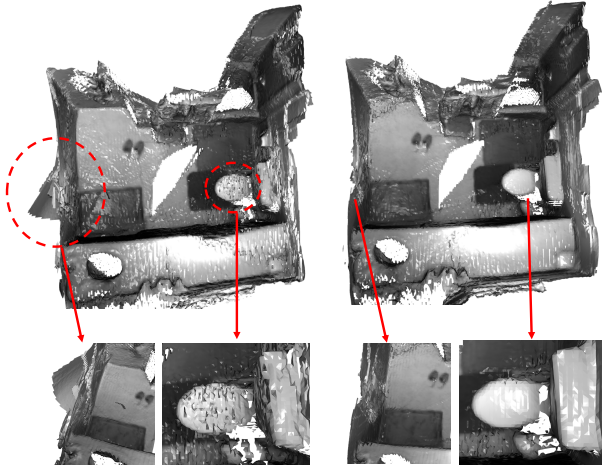


Fig. 10. TSDF fusion of predicted depth maps (**left**: without multi-view optimization, **right**: with multi-view optimization). As highlighted in the red circles, multi-view optimization helps to increase depth map accuracy and consistency.

TABLE I

DEPTH PREDICTION RESULTS OF THE SCANNET DATASET (IN METERS)

		0100_00	0406_02	0382_01	0084_00	0702_02	0406_01	0695_03	
ORB Points (sparse)	MAE	.028	.031	.068	.063	.032	.063	.072	
	RMSE	.065	.092	.134	.124	.087	.133	.120	
DeepFactors [13]	MAE	.185	.177	.358	.237	.368	.267	.291	
	RMSE	.220	.203	.432	.276	.397	.398	.408	
Ma <i>et al.</i> [8]	MAE	.141	.088	.140	.199	.123	.251	.272	
	RMSE	.285	.208	.284	.357	.235	.321	.337	
Ours	w/o rep error & multiview opt.	MAE	.050	.066	.123	.069	.069	.070	.086
		RMSE	.102	.140	.277	.170	.122	.141	.127
	w/o rep error	MAE	.044	.061	.116	.067	.061	.064	.079
		RMSE	.098	.141	.273	.165	.120	.139	.127
	w/o multiview opt.	MAE	.059	.068	.129	.089	.073	.069	.097
		RMSE	.110	.142	.266	.181	.123	.141	.143
	Full	MAE	.046	.060	.115	.073	.059	.063	.079
		RMSE	.100	.136	.260	.172	.113	.137	.125

2) *EuRoC MAV Dataset*: We also evaluate our method on the EuRoC MAV dataset [28] by running ORB-SLAM3 in visual-inertial mode. We generate ground truth depth maps by rendering the LiDAR data onto each frame. While the sequences from the ScanNet training set contain small rooms with furniture, the EuRoC dataset has much larger scenes with very different objects. We report reconstruction error results for different configurations of our system in Table II. These error values are higher compared to those obtained on the ScanNet dataset. We think this happens because the network learns the sizes and shapes of objects typically found in the apartments of the ScanNet dataset, while the environment in the EuRoC dataset is a large laboratory with technical equipment. However, compared to DeepFactors (which uses only intensity image conditioning), our method is less affected by the dataset domain change. This demonstrates that conditioning on sparse points improves the generalization capability of the network. Regarding reprojection error conditioning,

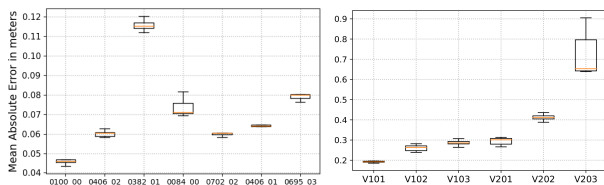


Fig. 11. Boxplot of our method (Full) on 2 datasets (**left**: ScanNet, **right**: EuRoC). The vertical axis represents Mean Absolute Error of depth prediction. We run our method 5 times on each sequence.

Fig. 12 shows an example of its importance for filtering out gross outliers. Outliers are more common in EuRoC as the dataset contains more texture-less walls and the 3D location of features is more difficult to calculate accurately. Fig. 11 shows errors over multiple trials on both ScanNet and EuRoC sequences to indicate the statistical significance of our results.

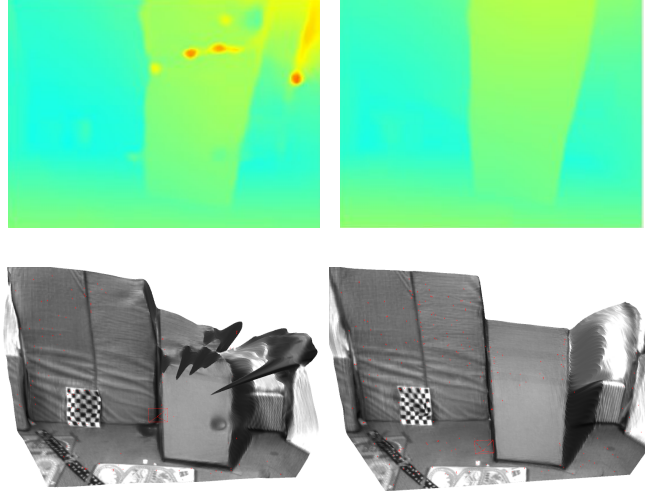


Fig. 12. A 2D/3D visualization of a depth prediction sample on the EuRoC dataset (**left**: without reprojection error conditioning, **right**: with reprojection error conditioning). Reprojection error conditioning helps in filtering out gross outliers.

TABLE II

DEPTH PREDICTION RESULTS OF THE EUROC DATASET (IN METERS)

		V101	V102	V103	V201	V202	V203	
ORB Points (sparse)	MAE	.115	.177	.216	.183	.291	.610	
	RMSE	.283	.381	.405	.374	.617	.801	
DeepFactors [13]	MAE	.842	.875	.833	.859	1.29	1.08	
	RMSE	1.05	1.03	.940	1.02	1.53	1.25	
Ma <i>et al.</i> [8]	MAE	.495	.509	.492	.486	.504	.871	
	RMSE	.598	.611	.595	.610	.660	1.08	
Ours	w/o rep error & multiview opt.	MAE	.280	.334	.354	.390	.533	.790
		RMSE	.435	.446	.460	.572	.782	1.14
	w/o rep error	MAE	.231	.289	.356	.336	.515	.780
		RMSE	.417	.419	.457	.551	.764	1.10
	w/o multiview opt.	MAE	.296	.358	.412	.373	.505	.725
		RMSE	.450	.479	.528	.566	.748	1.00
	Full	MAE	.192	.259	.283	.290	.415	.686
		RMSE	.381	.369	.407	.428	.655	.952

3) *Runtime measurement*: Table III shows the measured average time over the EuRoC and ScanNet dataset in milliseconds. Dense depth prediction times are reported per frame while multi-view optimization times are reported for formulations with 4 keyframes. We used the TensorFlow C++ API to run the trained model, but as Table III shows we found that running the densification network via the Python API is 20 times faster due to a better model loading procedure. We believe this speedup could also be replicated for our C++ system with an equivalent model loading process. Our approach can predict 4 dense depth images and refine them through multi-view optimization at a rate of around 1Hz. This frequency is sufficient for map updates as long as the camera motion is not aggressive.

4) *Limitations*: As we discussed in Sec. VI-A2, the performance of the network can be affected by large domain changes

TABLE III
MEAN TIMING RESULTS (MILLISECONDS).

Dense Prediction	235 (C++ API) 11 (Python API)
Multi-view Optimization	170

between training and testing data. This is mitigated to some extent by sparse points conditioning which acts as an online scale adjustment. Also, since we assume high quality points and camera poses given by the SLAM system, our method cannot handle scenes where SLAM fails due to low texture or very fast camera motion. However, as long as visual tracking can work, the SLAM system normally provides around 200 to 1000 points per-frame which is sufficient for depth completion to work well.

B. Qualitative Map Evaluation

1) *Local Mapping*: Our dense mapper continuously refines a keyframe window which consists of the latest keyframe and its top 3 covisibility keyframes by minimizing the error functions described in Section V. This process improves consistency for depth maps in overlapping frames.

We compare this procedure to Kimera [15], a real-time geometric system which generates meshes using 2D Delaunay triangulation and sparse SLAM points. Fig. 13 shows a comparison between maps generated by our system and those from Kimera. Using a network to learn about shapes and sizes of objects in indoor spaces leads to more accurate and smooth results compared to purely geometric approaches.

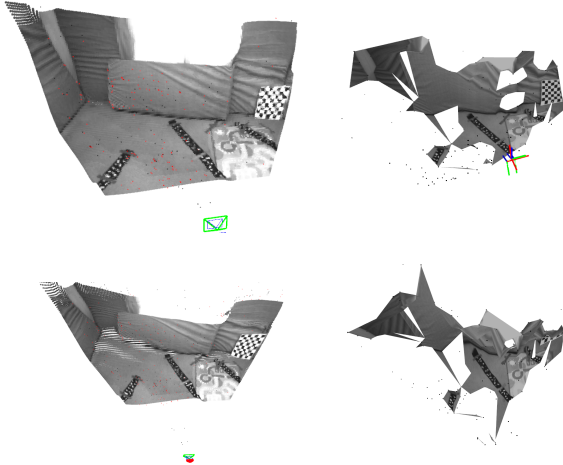


Fig. 13. An example of dense local mapping (**left**: ours, **right**: Delaunay meshing by Kimera [15]). While both methods rely on visual-inertial SLAM, our method generates more complete and smooth local maps.

2) *Globally Consistent Depth Map Fusion*: Since the SLAM thread (ORB-SLAM3) provides accurate and globally consistent camera poses, we can not only refine our dense prediction but also fuse depth maps into a globally consistent dense 3D model. Fig. 14 shows examples of offline volumetric TSDF fusion of depth maps from CodeMapping on scenes from the ScanNet test set. This is done after the SLAM system has processed the entire sequence. These 3D models are essential for tasks which require rich geometry information such as obstacle avoidance, AR/VR and physics simulation.

In Fig. 15 we carry out a physics simulation experiment by taking TSDF models generated with and without multi-view optimization and we simulate throwing a set of balls towards a wall. The example illustrates the practical importance of methods which generate not only dense but consistent depth maps.

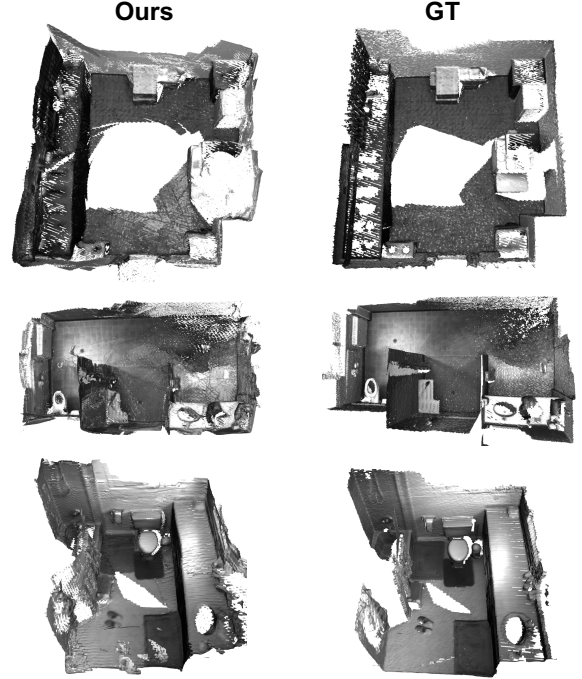


Fig. 14. Example 3D reconstructions on the ScanNet test set from our system (left) compared to ground truth (right). We generate globally consistent 3D models using TSDF fusion of depth maps from our system and camera poses from the SLAM system.

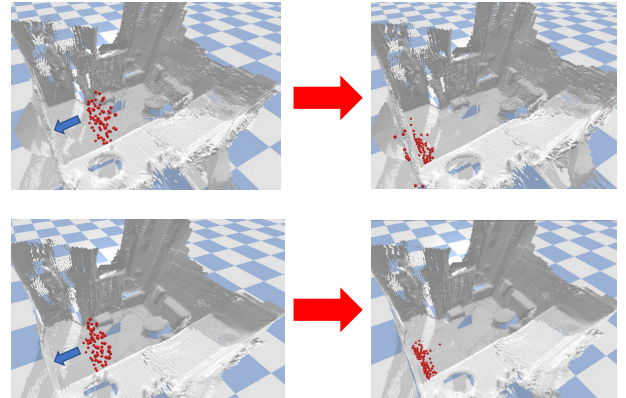


Fig. 15. A physics simulation example using the dense 3D model from TSDF fusion – generated without (**top**) and with (**bottom**) multi-view optimization, where red balls are thrown towards the wall. In the top 3D model the wall is not straight and the balls sink into it, but in the bottom model the balls bounce off the wall correctly thanks to accurate geometry enabled by multi-view optimization.

3) *Pure Monocular Dense Mapping*: Although the results presented so far were obtained using metric SLAM, our method could also be applied to a scale-less monocular SLAM result. We define the scale factor as the median of the depth maps of the training data divided by the median of the

depth maps from monocular ORB-SLAM. We multiply the depth maps and translational components of the camera poses provided by monocular SLAM by this scale factor and obtain 3D reconstructions of promising quality (Fig. 16).

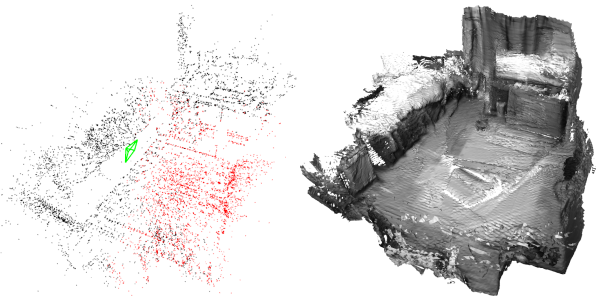


Fig. 16. Example 3D reconstruction from pure monocular ORB-SLAM (**left**: monocular SLAM result, **right**: our dense mapping result). The scene represents an apartment sequence which we manually collected.

VII. CONCLUSIONS

In this paper, we introduced a novel real-time multi-view dense mapping method which complements sparse SLAM systems. We first proposed a novel VAE architecture conditioned on image intensity, sparse depth and reprojection error maps, which performs uncertainty-aware depth completion. We also proposed an approach to integrate the VAE into an existing keyframe-based sparse SLAM system using multi-threading. This flexible integration allows us to run dense bundle adjustment without delaying the main SLAM process. We evaluate our system on two public datasets and demonstrate that our method performs better than other dense depth prediction methods, while multi-view optimization further improves accuracy and consistency. Our mapper provides not just locally but also globally consistent depth maps through fusion in a 3D global TSDF model. In future work, our mapper can be extended to use semantic segmentation and instance segmentation for more informative 3D mapping.

REFERENCES

- [1] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, “CodeSLAM — learning a compact, optimisable representation for dense visual SLAM,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [2] L. Platinsky, A. J. Davison, and S. Leutenegger, “Monocular visual odometry: Sparse joint optimisation or dense alternation?” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [3] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.
- [4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM,” *arXiv preprint arXiv:2007.11898*, 2020.
- [5] C. Houseago, M. Bloesch, and S. Leutenegger, “KO-Fusion: dense visual SLAM with tightly-coupled kinematic and odometric tracking,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.
- [6] R. A. Newcombe, S. Lovegrove, and A. J. Davison, “DTAM: Dense Tracking and Mapping in Real-Time,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.
- [7] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.
- [8] F. Ma and S. Karaman, “Sparse-to-dense: Depth prediction from sparse depth samples and a single image,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [9] Y. Zhang and T. Funkhouser, “Deep depth completion of a single RGB-D image,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [10] X. Xiong, H. Xiong, K. Xian, C. Zhao, Z. Cao, and X. Li, “Sparse-to-dense depth completion revisited: Sampling strategy and graph construction,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [11] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse, “MonoSLAM: Real-Time Single Camera SLAM,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29, no. 6, pp. 1052–1067, 2007.
- [12] G. Klein and D. W. Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” in *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.
- [13] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, “DeepFactors: Real-time probabilistic dense monocular SLAM,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.
- [14] A. I. Mourikis and S. I. Roumeliotis, “A multi-state constraint Kalman filter for vision-aided inertial navigation,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.
- [15] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, “Kimera: an open-source library for real-time metric-semantic localization and mapping,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [16] D. Zhou, Y. Dai, and H. Li, “Reliable scale estimation and correction for monocular visual odometry,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2016.
- [17] T. Schöps, T. Sattler, and M. Pollefeys, “BAD SLAM: Bundle adjusted direct RGB-D SLAM,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [18] K. Tateno, F. Tombari, I. Laina, and N. Navab, “CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [19] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, “D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [20] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof, “Image guided depth upsampling using anisotropic total generalized variation,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.
- [21] R. Cheng, C. Agia, Y. Ren, X. Li, and L. Bingbing, “S3CNet: A sparse semantic scene completion network for lidar point clouds,” *Conference on Robot Learning*, 2020.
- [22] Z. Murez, T. van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich, “Atlas: End-to-end 3d scene reconstruction from posed images,” 2020.
- [23] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, “DeepMVS: Learning multi-view stereopsis,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [24] X. Zuo, N. Merrill, W. Li, Y. Liu, M. Pollefeys, and G. Huang, “CodeVIO: Visual-inertial odometry with learned optimizable dense depth,” *arXiv preprint arXiv:2012.10133*, 2020.
- [25] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015.
- [26] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “ScanNet: Richly-annotated 3D reconstructions of indoor scenes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [27] F. Dellaert, “Factor graphs and GTSAM: A hands-on introduction,” Georgia Institute of Technology, Tech. Rep., 2012.
- [28] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The EuRoC micro aerial vehicle datasets,” *International Journal of Robotics Research (IJRR)*, vol. 35, no. 10, pp. 1157–1163, 2016.