

# UC Merced

## UC Merced Previously Published Works

### Title

Smart three-dimensional processing of unconstrained cave scans using small unmanned aerial systems and red, green, and blue-depth cameras

### Permalink

<https://escholarship.org/uc/item/9r73473c>

### Journal

International Journal of Advanced Robotic Systems, 19(2)

### ISSN

1729-8806

### Authors

Zhang, Guoxiang  
Moyes, Holley  
Chen, YangQuan

### Publication Date

2022-03-01

### DOI

10.1177/17298814211017728

Peer reviewed

# Smart three-dimensional processing of unconstrained cave scans using small unmanned aerial systems and red, green, and blue-depth cameras

*International Journal of Advanced  
Robotic Systems*

March-April 2022: 1–15

© The Author(s) 2022

Article reuse guidelines:

[sagepub.com/journals-permissions](https://sagepub.com/journals-permissions)

DOI: 10.1177/17298814211017728

[journals.sagepub.com/home/arx](https://journals.sagepub.com/home/arx)Guoxiang Zhang<sup>1</sup>, Holley Moyes<sup>2</sup>, and YangQuan Chen<sup>1</sup> 

## Abstract

This article focuses on a novel three-dimensional reconstruction system that maps large archeological caves using data collected by a small unmanned aircraft system with red, green, and blue-depth cameras. Cave sites often contain the best-preserved material in the archeological record. Yet few sites are fully mapped. Large caves environment usually contains complex geometric structures and objects, which must be scanned with long overlapped camera trajectories for better coverage. Due to the error in camera tracking of such scanning, reconstruction results often contain flaws and mismatches. To solve this problem, we propose a framework for surface loop closure, where loops are detected with a compute unified device architecture accelerated point cloud registration algorithm. After a loop is detected, a novel surface loop filtering method is proposed for robust loop optimization. This loop filtering method is robust to different scan patterns and can cope with tracking failure recovery so that there is more flexibility for unmanned aerial vehicles to fly and record data. We run experiments on public data sets and our cave data set for analysis and robustness tests. Experiments show that our system produces improved results on baseline methods.

## Keywords

3D reconstruction, fast surface loop detection, surface loop filtering, loop optimization

Date received: 17 December 2020; accepted: 21 April 2021

Topic: Field Robotics

Topic Editor: Nak-Young Chong

Associate Editor: Grazia Cicirelli

## Introduction

It has long been recognized that cave sites often contain the best-preserved material in the archeological record. Cave archeology has developed its methodologies for mapping and recording sites, yet few sites are mapped to true three-dimensional (3D) models because it is a slow and tedious process for archeologists to record and bookkeep caves. They need to incrementally set up baseline along the cave and then measure the distance from the baseline to cave walls or objects of interest and mark walls or objects in a two-dimensional (2D) map by hand.<sup>1</sup> This slow process has a major negative impact on cultural relic preservation. Typically, archeological teams will visit a site and begin

to record it in 1 year, but when they come back to finish data collection, it has been looted, artifacts stolen, architecture destroyed, and the archeological record disturbed.

<sup>1</sup> Mechatronics, Embedded Systems and Automation (MESA) Lab, University of California, Merced, CA, USA

<sup>2</sup> School of Social Sciences, Humanities, and Arts, University of California, Merced, CA, USA

### Corresponding author:

YangQuan Chen, Mechatronics, Embedded Systems and Automation (MESA) Lab, University of California, 5200 N Lake Rd, Merced, CA 95343, USA.

Email: [ychen53@ucmerced.edu](mailto:ychen53@ucmerced.edu)



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<https://creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without

further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

Therefore, archeologists need a faster, more efficient method of surveying and recording the sites.

To accelerate cave mapping, a system that can automate the process needs to be developed. In this system, we first focus on building globally consistent and accurate 3D models using red, green, and blue-depth (RGB-D) data recorded with unmanned aerial vehicles (UAVs). This falls into the research area of visual simultaneous localization and mapping (vSLAM). There are still many challenges in making a well-working vSLAM system that emphasizes surface estimate accuracy.<sup>2,3</sup> Major difficulties are from three sources. First, the camera tracking error accumulation problem can be considered unavoidable due to the incremental nature of any vSLAM system. Second, it is hard to find all the important loops. Intuitively, the more loops in data, the more information can be used to recover precise camera trajectory and the 3D model. But, in practice, when running existing vSLAM systems on data sets with loopy motion at different scales, mismatches can always be found when scanned more than once. This means that loops are not successfully detected, indicating that the recovered camera trajectory is not precise enough. Third, an effective and optimal way of optimizing loop closure in a dense vSLAM system is not yet attempted, due to perhaps the cost of reintegrating dense models.

To get better camera tracking and dense mapping accuracy, researchers tried different ways. In Lefloch et al.,<sup>4</sup> curvature information was added into the frame-to-model iterative closest point (ICP). To detect those loops, most vSLAM systems<sup>5-7</sup> use bag of words (BoW),<sup>8</sup> but it is well known that it is not very reliable under lighting conditions or viewing angle changes. Since the BoW only matches sparse features from images but cannot fully utilize all camera observation data and spatial information. On the other hand, vSLAM systems tend to add the loops very conservatively to reduce the severe influence of the false loops, thus many important loops may not be connected. Even after loops are successfully detected, there is still another problem in the dense vSLAM system: how to correct reconstructed surface optimally. Since most dense vSLAM systems<sup>9,6,10</sup> use a frame-to-model fusion process, which makes it difficult to quantify, isolate, and remove the influence of past camera data, and it is also computational expensive for a full camera data sequence refusion. Whelan et al.<sup>10</sup> suggested forming a deformation graph across the reconstructed dense model to deform its surface to connect the loop. When the loop area is large, the model may not be deformed optimally, since past camera observations are not reused to manipulate the 3D model. They assumed that the scenes are elastic, but in reality, they are mostly rigid.

Motivated by the fact that humans can notice mismatches in 3D models very easily by looking at the spatial displacement of surfaces. We propose to resolve mismatches directly by closing surface loops to get a consistent 3D model and a precise camera trajectory estimate in the vSLAM system. After surface loops are detected, instead of

optimizing surface directly to propagate correction introduced by surface loop, in this article, the surface loop correction is done through sparse feature bundle adjustment (BA), so that all the past camera poses can be corrected based on their observations. By running extensive experiments on different data sets, we observed that combining sparse features with surface loop closure can produce better results. Not only 3D models get improved, but also camera trajectories estimate becomes more accurate. This is because our framework can detect loops in the dense surface domain and optimize loops in the sparse feature domain. Note that our framework can detect surface loops, yet other means of detecting loops can still be utilized.

In the following, we summarize the key contributions of our method:

1. We propose a novel 3D reconstruction system that corrects surface loops with sparse feature-based BA. We demonstrate that this novel system can give much-improved camera tracking and dense modeling results.
2. We propose a fast 3D surface-based loop detection method, which is based on a new compute unified device architecture (CUDA)-accelerated point cloud registration algorithm. Experiments show that it is fast and accurate.
3. We propose a novel objective function for surface loop filtering with a sparse feature-based optimization graph. This graph is more robust to different scan patterns and can cope with tracking failure and recovery so that there is more flexibility for UAVs to fly and record data. In addition to the flexibility, experiments show that it performs better than state-of-art methods when only a limited number of loops are detected.

## Related work

### *Related work in visual SLAM*

Visual SLAM has been studied actively by researchers from different fields, such as robotics, computer vision, and computer graphics. They solve this problem with their emphases and preferences, which lead to diverse visual SLAM systems. Sparse feature-based SLAM systems are well developed because sparse features can be used to downsample data from sensor reading (e.g. images) to sparse data representation as image keypoints and feature vectors, which means less computation since data from different frames are matched solely based on feature vectors of their keypoints. Extended Kalman filter or particle filter-based filtering approaches<sup>11,12</sup> can take keypoints as visual landmarks and solve vSLAM as a data filtering process. A drawback of this approach is that the filter cannot be reoptimized again based on all previous data. Then, maximum a posteriori (MAP)-based approaches are used

to optimize all observed camera data in a batch setting,<sup>7</sup> which utilizes BA from Structure from Motion,<sup>13,14</sup> and to get better accuracy.<sup>7</sup> To run BA for loop closure, the loops need to be detected. In the sparse image feature setting, BoW-based loop closure is widely used. But it gives a high portion of false loops, which can severely degrade the performance of a vSLAM system, so a very strict loop filtering is often used,<sup>7</sup> where many loops are rejected. This causes a big problem when there are many loopy motions in camera movement.

Another line of vSLAM research focuses on surface reconstruction. With the parallel processing power of graphics processing unit (GPU), Newcombe et al. proposed KinectFusion<sup>9</sup> which performs real-time dense 3D camera tracking and model fusion. It has a volumetric scene representation, which can be rendered to a depth map at a given camera pose. Tacking is done through a frame-to-model projective ICP, which is parallelized on GPU for real-time performance. Finally, new camera data are fused into the volumetric model using a running average. KinectFusion can be considered of fusing very local loops together using the model it maintains as a proxy, but it does not close large loops. To close large loops, it is important to detect loops and find relative poses between loop areas. In BoW, image keypoints and features are used for both loop detection and relative pose generation, but in dense 3D systems, there is no such comparably reliable point cloud feature. Whelan et al. use BoW in a dense SLAM system called Kintinuous,<sup>6</sup> which is an extended KinectFusion system. Later, to better solve the loop detection and optimization problem, ElasticFusion<sup>10</sup> proposed to use ICP to find relative poses of potential loops, which are proposed by two sources of information: spatial prior and appearance-based place recognition. Our work shares similarities with Whelan et al.,<sup>10</sup> but we propose a different approach and underlying algorithms, instead of using projective ICP, which highly depends on initialization. We propose a GPU-based global point cloud registration method to detect loops with other prior information from sparse feature alignment. After surface loop was detected, in Kintinuous, a pose graph of keyframes was utilized, while the authors mentioned that mesh deformation was required to get smooth 3D models, which indicates loop correction is not done optimally. In ElasticFusion, the pose graph is replaced by a deformation graph distributed inside the dense model. This deformation graph does not have a backing physical meaning, because most of the scenes scanned are not elastic. In our framework, we utilize BA to have a MAP correction of all past keyframes, which is theoretically optimal. BundleFusion<sup>14</sup> used BA to optimize loops, but they do not close surface loops. Instead, they close sparse feature loops and only use the dense surface for feature correspondence search and tracking.

## Related work in loop detection

Handcrafted image features are widely used in loop closure detection. Cummins and Newman propose Fast appearance based mapping (FAB-MAP),<sup>15</sup> which is a probabilistic framework for navigation using only appearance data. Angeli et al. propose an online method to run visual word-based loop detection within the framework of an online image retrieval task.<sup>16,17</sup> Williams et al.<sup>18</sup> describe a relocalization module, in which relocalization is performed by a randomized lists classifier to establish landmark correspondences in the image and then random sample consensus (RANSAC) to determine the pose robustly from these correspondences. Another widely used approach, distributed BoW,<sup>8</sup> is proposed by Galvez-Lopez and Tardos, which use BoW for visual place recognition with features from accelerated segment test (FAST) keypoint detector and Binary Robust Independent Elementary Features (BRIEF) with a binary vocabulary tree.

There are new attempts to make further improvements, including modifications on the BoW<sup>19-21</sup> and detection by image sequences.<sup>22-24</sup> There are also explorations to detect loops with more than a single image.<sup>25,23,26</sup> Some works try to improve loop detection precision with loop verification threshold learning in RANSAC based on geometric.<sup>27</sup> Some others try to make improvements for special environment.<sup>28,29</sup>

After, convolutional neural networks (CNNs) dominate computer vision-related tasks<sup>6</sup>. There are research works that try to build loop detection methods with CNN features for similarity search.<sup>30-32,26</sup> Another line of work builds special CNN architectures for loop detection problem.<sup>33,33</sup>

## Overview of the proposed system

The method proposed aims at producing accurate 3D models by detecting and optimizing surface loops in sparse feature-based visual SLAM systems. By adding surface loop closure into a vSLAM system, we can get globally consistent and optimal 3D models. With our proposed algorithms, we build a novel system with five components: (1) tracking, (2) surface model fusion, (3) fast surface loop detection, (4) surface loop filtering, and (5) loop optimization.

### Tracking

We employ the tracking part from ORB-SLAM2,<sup>7</sup> which is a very wellimplemented sparse feature-based SLAM system. Inside this tracking module, the Oriented FAST and Rotated BRIEF (ORB) features are extracted for keypoint matching. Then frames are tracked against keyframes with motion estimate and then refined with a local sparse map. Keyframes are generated when tracking is weak, or the local BA thread is free. Local BA is used to correct the reprojection error of feature correspondences among

covisible keyframes in a background thread. This tracking module provides camera poses for each frame and a covisibility graph across keyframes.

### Surface model fusion

We fuse surface models on a GPU using surfels as a map representation similar to the literatures.<sup>34,10</sup> Each surfel has a position  $\mathbf{p}$ , normal vector  $\mathbf{n}$ , radius  $r$  and confidence  $c$ . We fuse keyframes within every  $k$  frames ( $k = 50$  for all experiments) into a surface fragment. These surface fragments are generated for two reasons. One is to integrate out raw RGB-D data noise. Another one is to reduce the number of 3D pieces, so that loop detection computation is accelerated. After each scene fragment is generated, for the later optimization process, it is linked to the keyframe whose timestamp is closest to the first frame within the range.

### Fast surface loop detection

The proposed surface loop detection is done by point cloud registration on surface fragments. To detect surface loops effectively, the covisibility graph from tracking is utilized to prefilter covisible surface fragment pairs that are already connected. Thus, a majority of unnecessary computation can be avoided. To detect surface loops efficiently, we propose using CUDA to accelerate point cloud registration. Details of this acceleration are described in the next section.

### Surface loop filtering

After loop candidates are detected, they need to go through a novel loop verification algorithm, so that false loops would not diverge the subsequent optimization process. This is in the surface loop filtering section.

### Loop optimization

After surface loops are detected and verified, the loop pairs are used to connect pose graph vertices and also trigger more image loop detection, which again uses spatial prior and ORB feature matching. Then, the pose graph is optimized to give a coarse pose correction and then a full BA is performed to get MAP optimally fine-tuned camera trajectory estimate. Details are in the loop optimization section.

### Fast surface loop detection

To formulate surface loop detection formally, we denote a surface fragment as a set of points  $\mathbf{P}$  with their normal. Then the surface fragment-based loop detection problem can be solved by point cloud registration methods. In a 3D reconstruction system, it is desired to get results quickly. In ElasticFusion, Whelan et al. resort to projective ICP, which can be performed very quickly on GPU. But, when surface mismatches are big, the initialization-dependent nature of ICP makes it difficult to converge to the right solution. So we turn

to global point cloud registration in our framework, which does not depend on initial alignment at all. Our algorithm is based on RANSAC and inspired by Choi et al.,<sup>35</sup> which uses a method that is modified from Point Cloud Library.<sup>36</sup> The major differences are that we accelerated the most time-consuming parts using GPU programming with an efficient nearest neighbor (NN) search method. Traditionally, RANSAC is formulated as an iterative process with proved convergence.<sup>37</sup> But different iterations and different hypotheses can be considered to be totally independent of each other. This means different hypotheses can be mapped to different processing cores to be tested in parallel.

As shown in Algorithm 1, point clouds are downsampled to the resolution of the typical precision of RGB-D sensors to reduce unnecessary computation. Fast point feature histograms (FPFH) features are extracted for each point in  $\mathbf{P}$  and  $\mathbf{Q}$  for point correspondence pairs generation. To make the NN search more efficient, we precached all the NNs of  $\mathbf{P}$  in  $\mathbf{Q}$  using FPFH feature distance. Then, for each hypothesis, four points are randomly sampled from  $\mathbf{P}$ , and their correspondences are found through the precached NNs. After that, a pre-rejection step, which rejects hypotheses whose point pairs cannot make a similar polygon, is performed.  $\tau$  is a similarity threshold and set to 0.9 in all our experiments. Then, hypotheses testing, which is the most time-consuming step, tests both inlier ratio and fitness score on GPU. We test each hypothesis on a thread block with efficient parallel reductions. For the NN search during hypotheses testing, we utilized a 3D grid to replace the k-d tree to fit the special need of GPUs, since a GPU will slow down when different threads go to different code branches during the k-d tree search. We propose using a 3D grid for the NN search, given that the point clouds to be searched can be stored in a limited volume. This guarantees that we can use a grid with a limited size for the NN search without jeopardizing search accuracy. When a point is stored into the search grid, the indices  $(i_x, i_y, i_z)$  of its cell are calculated by the following equation

$$\begin{aligned} i_x &= (x_p - x_c)/l_c \\ i_y &= (y_p - y_c)/l_c \\ i_z &= (z_p - z_c)/l_c \end{aligned} \quad (1)$$

where  $l_c$  is the cell edge length of the NN 3D grid. The  $(x_c, y_c, z_c)$  is the coordinate of the center of target point cloud. It is subtracted so that the translation of the point cloud does not affect searching. When a point wants to query its NN, the search is accomplished through a table looking up, which has a time complexity of  $\mathcal{O}(1)$ , given cell edge length the same as point cloud downsample resolution. It is faster than the k-d tree which has a  $\mathcal{O}(\log n)$  time complexity. We observed speedup by only replacing the k-d tree with a 3D search grid in CPU only code. More importantly, there is no branching during searching, so it fits much better on a GPU than the k-d tree.

**Algorithm 1.** Global Point Cloud Registration on GPU

---

**Input:** A pair of point cloud  $P$  and  $Q$   
**Output:**  $T_{QP}$  if  $P$  and  $Q$  can be aligned together

Downsample  $P$  and  $Q$ ;  
 Compute FPFH features  $F(P)$  and  $F(Q)$ ;  
 $T_{QP} \leftarrow I$ ;  
 $feature\_NN\_cache \leftarrow \emptyset$   
 Quadruple point set samples  $S_P \leftarrow \emptyset, S_Q \leftarrow \emptyset$

// Parallel FPFH feature pre-matching  
**for**  $i \leftarrow 1$  **to**  $P\_count$  **do on GPU cores in parallel**  
    $feature\_NN\_cache[i] \leftarrow$  NN of  $F(p_i)$  in  $F(Q)$ ;  
**end for**

// Randomly sample hypotheses on GPU  
**for**  $i \leftarrow 1$  **to**  $sample\_number$  **do on GPU cores in parallel**  
    $S_P[i] \leftarrow$  randomly picked  $(p_0, p_1, p_2, p_3)$  from  $P$  ;  
   **for all**  $(p_0, p_1, p_2, p_3)$  **do**  
      $q_j \leftarrow$  NN of  $p_j$  in  $Q$  using  $feature\_NN\_cache[j]$   
   **end for**  
    $S_Q[i] \leftarrow (q_0, q_1, q_2, q_3)$   
   // Hypotheses pre-rejection  
   **for all**  $(p_0, p_1, p_2, p_3), (q_0, q_1, q_2, q_3)$  **do**  
     **if**  $\|p_k - p_{k+1}\| < \tau \|q_k - q_{k+1}\|$  or vice versa  
   **then**  
     set current hypothesis pre-rejected in  $S_P$  and  $S_Q$ ;  
   **end if**  
   **end for**  
**end for**

Stream compact non-rejected hypotheses;

// Consensus for the remaining quadruple pairs  
**for**  $i \leftarrow 1$  **to**  $remain\_number$  **do on GPU cores in parallel**  
   Estimate transformation  $T_{QP}$  from  $S_P[i]$  to  $S_Q[i]$  ;  
   Find all inliers under  $T_{QP}$  hypothesis;  
    $test\_log \leftarrow$  inlier ratio, fitness score  
**end for**

// Get final result from  $test\_log$   
 Find the Hypothesis that has max inlier ratio and fitness score passing minimum threshold in  $test\_log$ .  
**if** Hypothesis found **then**  
   return its  $T_{QP}$ ;  
**end if**

---

**Surface loop detection evaluation**

We run experiments to compare speed, recall, and precision performance with the baseline method: point cloud registration part of Choi-Zhou-Koltun (CZK) on redwood pairwise registration evaluation data set by Choi *et al.* We report results in Table 1. Recall and precision for CZK are from reports of article authors, but speed is measured on our machine. We use the same hyper-parameter values as in the published code of CZK: 0.05 m as point cloud down-sampling leaf size, 0.1 m as normal estimate radius, 0.25 m as FPFH feature estimate radius, 4,000000 as hypotheses count, and 0.075 m as maximum point correspondence distance. We use an Intel i7-6850 K (<https://ark.intel.com/content/www/us/en/ark/products/94188/intel-core-i7-6850k-processor-15m-cache-up-to-3-80-ghz.html>) clocked at 3.6 GHz and an NVIDIA Titan X Pascal (<https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal/>) for our evaluation.

**Observations.** Our global point cloud registration can finish in around 20 ms, which is around 366 times faster than CZK as in Table 1. With this speed, it can run at 50 Hz, which means we can process more loop candidates. Our recall performance is not compromised for speed. Instead, it is even better than all these two methods. From Table 1, the low precision is an issue that needs to be solved, so an effective loop filtering method is proposed and described in the following section.

**Surface loop filtering**

The results out of the surface loop detection algorithm in the previous section have a low precision problem. Choi et al. proposed to use a line process-based optimization to solve it.<sup>35</sup> But that method requires scanning to be tracked fully successfully from the beginning to the end, and each surface fragment cannot be empty. However, during a long scanning session, it is almost impossible to guarantee that RGB-D cameras always have surfaces observed within their effective range. A failure to maintain that will lead to an empty surface fragment. Thus, the graph vertices are potentially divided into more than one subgraph, which leads to erroneous optimization results. A similar problem will occur when tracking failure is present in the tracking part, which will break the optimization graph as well.

**Table 1.** Performance evaluation of different point cloud registration methods.

	Time/pair (ms)		Recall		Precision	
	CZK	Ours	CZK	Ours	CZK	Ours
Living room 1	7606.20	<b>24.71</b>	61.2%	<b>62.4%</b>	<b>27.2%</b>	23.6%
Living room 2	7469.58	<b>18.19</b>	49.7%	<b>50.3%</b>	17.0%	<b>20.5%</b>
Office 1	7556.02	<b>21.63</b>	64.4%	<b>67.8%</b>	19.2%	<b>19.3%</b>
Office 2	7418.12	<b>17.45</b>	61.5%	<b>71.1%</b>	14.9%	<b>16.4%</b>
Average	7512.23	<b>20.50</b>	59.2%	<b>62.9%</b>	19.6%	<b>20.0%</b>

The best performance is marked in bold.

To address this problem, we propose to minimize (2) instead, which has a supporting optimization graph that is always fully connected whenever sparse features are matched from either tracking or failure recovery

$$\begin{aligned} \min_{\mathbb{X}, \mathbb{T}, \mathbb{S}} \quad & \sum_c \sum_{v \in V(c)} \|\tilde{\mathbf{x}}_v^c - \mathbf{K} \mathbf{T}_{cw} \mathbf{X}_v\|_{\Sigma} \\ & + \sum_{ij} f(\mathbf{T}_{iw}, \mathbf{T}_{jw}, s_{ij} | \mathcal{K}_{ij}) \quad (2) \\ \text{s.t.} \quad & 0 < s_{ij} < 1, \text{ for all } i, j \text{ pairs} \end{aligned}$$

where

$$\begin{aligned} & f(\mathbf{T}_{iw}, \mathbf{T}_{jw}, s_{ij} | \mathcal{K}_{ij}) \\ = & \sum_{(p_i, p_j) \in \mathcal{K}_{ij}} \left\| \Psi(s_{ij}) \cdot (\mathbf{p}_i - \mathbf{T}_{iw} \mathbf{T}_{jw}^{-1} \mathbf{p}_j) \right\|^2 \quad (3) \\ & + \|\gamma_{ij} - s_{ij}\|_{\Omega_{ij}}^2 \end{aligned}$$

The objective function optimizes over a set of point clouds  $\mathbb{X} = \{\mathbf{X}_v \in \mathbb{B}^3\}$  estimated from sparse feature key points, camera trajectory  $\mathbb{T} = \{\mathbf{T}_{cw} \in SE(3)\}$ , and a set of switch variables  $\mathbb{S} = \{s_{ij} \in \mathbb{R}\}$ . The first term in eq. (2) builds a least-square optimization graph for BA from sparse keypoint observations, where  $\mathbf{X}_v$  is the 3D coordinate of a point visible in the  $c$ th camera.  $\tilde{\mathbf{x}}_v^c$  is the 2D pixel observation coordinate of the 3D point  $\mathbf{X}_v$ .  $\mathbf{K}$  is the camera intrinsics matrix.  $\Sigma$  is the covariance matrix associated to the scale of the keypoint. The second term is surface loop connections with switchable constraints<sup>38</sup> with  $s_{ij}$  as a switch variable for surface loop connection that connects keyframes  $i$  and  $j$ . Let  $P_i$  be the surface fragment referred by keyframe  $i$ ,  $\mathcal{K}_{ij}$  is the set of NN correspondence pairs between  $\mathbf{T}_{iw}^{-1} \mathbf{P}_i$  and  $\mathbf{T}_{jw}^{-1} \mathbf{P}_j$  that are within distance  $\varepsilon = 0.05$  m, which is typical noise level of RGB-D sensor.

In eq. (3),  $\Psi(s_{ij})$  is a switch function and we use a linear function  $\Psi(s_{ij}) = s_{ij}$  as suggested in Sunderhauf and Protzel.<sup>38</sup>  $\Omega_{ij}$  is information of switchable prior constraints. It controls the influence of the loop candidate tested. Let  $\mathbf{T}_{ij}$  be the transformation that aligns all the  $\mathbf{p}_j$  to related  $\mathbf{p}_i$  in  $\mathcal{K}_{ij}$ , that is

$$\mathbf{p}_i \approx \mathbf{T}_{ij} \mathbf{p}_j, \text{ for all } (\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{K}_{ij}. \quad (4)$$

Then

$$\begin{aligned} & \sum_{(p_i, p_j) \in \mathcal{K}_{ij}} \left\| \Psi(s_{ij}) \cdot (\mathbf{p}_i - \mathbf{T}_{iw} \mathbf{T}_{jw}^{-1} \mathbf{p}_j) \right\|^2 \\ \approx & \sum_{(p_i, p_j) \in \mathcal{K}_{ij}} \left\| \Psi(s_{ij}) \cdot (\mathbf{p}_i - \mathbf{T}_{iw} \mathbf{T}_{jw}^{-1} \mathbf{T}_{ij}^{-1} \mathbf{p}_i) \right\|^2 \quad (5) \end{aligned}$$

To accelerate computation of eq. (5), we follow an approximation proposed in Choi et al.<sup>35</sup> The local parameterization of  $\mathbf{T}_{iw} \mathbf{T}_{jw}^{-1} \mathbf{T}_{ij}^{-1}$  is represented with a six-dimensional vector  $\xi = (\boldsymbol{\omega}, \mathbf{t})^T = (\alpha, \beta, \gamma, x, y, z)^T$ , which consists of three rotational angles  $\alpha, \beta, \gamma$  and three translation

components  $x, y, z$ . Since its rotation is small under the assumption that the registration result is good and camera pose estimates are not far away from the correct solution, the approximations  $\sin(\theta) \approx \theta$  and  $\cos(\theta) \approx 1$  are utilized for all  $\alpha, \beta, \gamma$ . Then

$$\begin{aligned} \mathbf{T}_{iw} \mathbf{T}_{jw}^{-1} \mathbf{T}_{ij}^{-1} \mathbf{p}_i & \approx \begin{bmatrix} 1 & -\gamma & \beta & x \\ \gamma & 1 & -\alpha & y \\ -\beta & \alpha & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p}_i \\ & = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \mathbf{p}_i + \mathbf{t} \quad (6) \\ & = \left( \mathbf{I} + \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix} \right) \mathbf{p}_i + \mathbf{t} \\ & = \mathbf{p}_i + \boldsymbol{\omega} \times \mathbf{p}_i + \mathbf{t} \\ & = \mathbf{p}_i - \mathbf{p}_i \times \boldsymbol{\omega} + \mathbf{t}. \end{aligned}$$

Plug eq. (6) into eq. (5)

$$\begin{aligned} & \sum_{(p_i, p_j) \in \mathcal{K}_{ij}} \left\| \Psi(s_{ij}) \cdot (\mathbf{p}_i - \mathbf{T}_{iw} \mathbf{T}_{jw}^{-1} \mathbf{p}_j) \right\|^2 \\ & \approx \sum_{(p_i, p_j) \in \mathcal{K}_{ij}} \left\| \Psi(s_{ij}) \cdot (-\mathbf{p}_i \times \boldsymbol{\omega} + \mathbf{t}) \right\|^2 \quad (7) \\ & = \sum_{(p_i, p_j) \in \mathcal{K}_{ij}} \left\| \Psi(s_{ij}) \cdot [-[\mathbf{p}_i]_{\times} | \mathbf{I}] \cdot \xi \right\|^2 \\ & = \left\| \Psi(s_{ij}) \cdot \xi \right\|_{A_L}^2, \end{aligned}$$

where

$$A_L^2 = \sum_{(p_i, p_j) \in \mathcal{K}_{ij}} [-[\mathbf{p}_i]_{\times} | \mathbf{I}]^T \cdot [-[\mathbf{p}_i]_{\times} | \mathbf{I}] \quad (8)$$

where

$$[-[\mathbf{p}_i]_{\times} | \mathbf{I}] = \begin{bmatrix} 0 & z_{pi} & -y_{pi} & 1 & 0 & 0 \\ -z_{pi} & 0 & x_{pi} & 0 & 1 & 0 \\ y_{pi} & -x_{pi} & 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

where  $x_{pi}, y_{pi}, z_{pi}$  are the three components of  $\mathbf{p}_i$ . Then our eq. (3) becomes

$$f(\mathbf{T}_{iw}, \mathbf{T}_{jw}, s_{ij} | \mathcal{K}_{ij}) = \left\| \Psi(s_{ij}) \cdot \xi \right\|_{A_L}^2 + \|\gamma_{ij} - s_{ij}\|_{\Omega_{ij}}^2. \quad (10)$$

After optimizing eq. (2), keyframes  $i$  and  $j$  should be connected as a loop if the optimized value of switch variable  $s_{ij}$  is greater than a threshold.

### Surface loop filtering evaluation

**Experiment design.** To understand the performance of the proposed method and compare it with CZK, we run experiments

**Table 2.** Performance evaluation of different loop filtering methods reported in percentage.

Recall (%) / Precision (%)	All pairs			Only non-covisible pairs		
	Registration	CZK	Ours	Registration	CZK	Ours
Living room 1	61.2/27.2	<b>57.6/95.1</b>	43.5/93.7	48.8/13.0	<b>48.8/92.2</b>	36.4/91.7
Living room 2	49.7/17.0	<b>49.7/97.4</b>	47.1/97.3	30.2/5.2	<b>30.2/94.1</b>	28.3/93.8
Office 1	64.4/19.2	<b>63.3/98.3</b>	34.4/ <b>98.4</b>	27.0/2.3	<b>27.0/62.5</b>	<b>27.0/100.0</b>
Office 2	61.5/14.9	<b>60.7/100.0</b>	58.5/96.3	29.3/2.9	<b>22.0/100.0</b>	<b>29.3/92.3</b>
Average	59.2/19.6	<b>57.8/97.7</b>	45.9/96.4	33.8/5.9	<b>32.0/87.2</b>	<b>30.3/94.5</b>

The best performance is marked in bold.

on the augmented ICL-NUIM data set for loop filtering evaluation. In the experiments, RGB-D frames are first fused into scene fragments with the implementation provided by CZK. Point cloud registration results from CZK are used as loop detections. Two sets of experiments are conducted: One takes all the successful point cloud registration results as loop detections, denoted as All pairs in Table 2. Another set of experiments only consider point cloud pairs that are not covisible in ORB-SLAM2 tracking results as loop detections, which is denoted as Only non-covisible pairs in Table 2. A pair of scene fragments is covisible if there are any covisible frames between two sets of frames contained in the two scene fragments. This only non-covisible set of loops is more close to practical use cases because the covisible pairs can be ignored for computational speed consideration and have been well connected in SLAM systems, already.

*Observations.* When only non-covisible pairs are presented, the proposed method outperforms CZK in average precision, while recall is only 2% less. For office 1 sequence, our method gives better results in both precision and recall, while CZK output only 62.5% precision. Such a low precision usually causes serious problems because too many false loops go in the loop optimization step. For the other three sequences, our method gives competitive results. When all pairs are considered, CZK performs so well that our method is closely under it. The difference is mainly in recall, while precision difference is very small. Considering precision is more important than recall for loop optimization, we would like to note that this minor difference rarely impacts on the final loop optimization result.

In addition to the cases where our method performs better, note that our method is less strict to use in practical scenarios, especially on a fast-moving UAV platform. Because our method does not require maintaining RGB-D cameras facing surfaces all the time, which is required by CZK. This requirement difference is inherently implied by underlying optimization pose graphs.

## Loop optimization

After a surface loop passing verification, map optimization is followed to reduce mismatches and errors. We employ pose graph optimization and new data association and finally run a

full BA to get a MAP-based optimization to correct the camera trajectory estimate and thus improve the 3D model.

When a surface loop pair  $\{i, j\}$  passes loop verification, we then try to find data association in the sparse feature domain. This is done by first retrieve all the covisible keyframes for both keyframes  $i$  and  $j$  noted as  $\mathcal{F}_i$  and  $\mathcal{F}_j$ . Then collect all the local sparse map points  $\{X_p\}_i$  and  $\{X_p\}_j$  that are observed by  $\mathcal{F}_i$  and  $\mathcal{F}_j$ , respectively. Then data association between  $\{X_p\}_i$  and  $\{X_p\}_j$  are constructed by both distances in image feature space and Euclidean space. After that, we run RANSAC to filter out outliers in the matches and also improve the transformation  $T_{ij}$ . If this process converges with enough inlier matches, we add a loop connection for keyframes  $i$  and  $j$ . And update keyframe connections by merging the observations of matched sparse map points. These merged map points will help improve during the final BA process. If it does not converge, we still add a loop connection with the  $T_{ij}$ . We do this because in some cases, even though the areas related to keyframes  $i$  and  $j$  do not have enough sparse map points for a good matching result, a loop connection can lead to a better initial start point for BA.

Finally, a full BA is performed for all the keyframes and observed map points by optimizing the following equation

$$\min \sum_c \sum_{v \in V(c)} \left\| \tilde{x}_p^c - \mathbf{K} T_{cw} \mathbf{X}_v \right\|_{\Sigma} \quad (11)$$

## Evaluation of the full 3D reconstruction system

In addition to experiments in the fast surface loop detection section and surface loop filtering section, extensive experiments are performed to evaluate the proposed full 3D reconstruction system on multiple data sets: our Maya cave data set, augmented Imperial College London and National University of Ireland Maynooth (ICL-NUIM),<sup>35</sup> Technical University of Munich (TUM) RGB-D data set,<sup>39</sup> Scene understanding 3D (SUN3D) data set,<sup>40</sup> and some other public data sequences. Comparisons are made with other online and offline methods. There are many SLAM algorithms and implementations. Here, we choose baseline methods in a way that they can best show the characteristics



of our proposed system: We choose ORB-SLAM2<sup>7</sup> since we use the tracking part of it and offline method CZK<sup>35</sup> because our loop filtering part is inspired by it. We denote the tracking part of ORB-SLAM2 as tracking and full ORB-SLAM2 as ORB-SLAM2 in all experiments. Results of baseline methods are from original articles or their authors when available.

### Maya cave data set

The Maya cave data set is a data set collected by a team of archeologists using an RGB-D sensor Kinect V1 in Maya caves at Las Cuevas, Belize. Light-emitting diode lights are

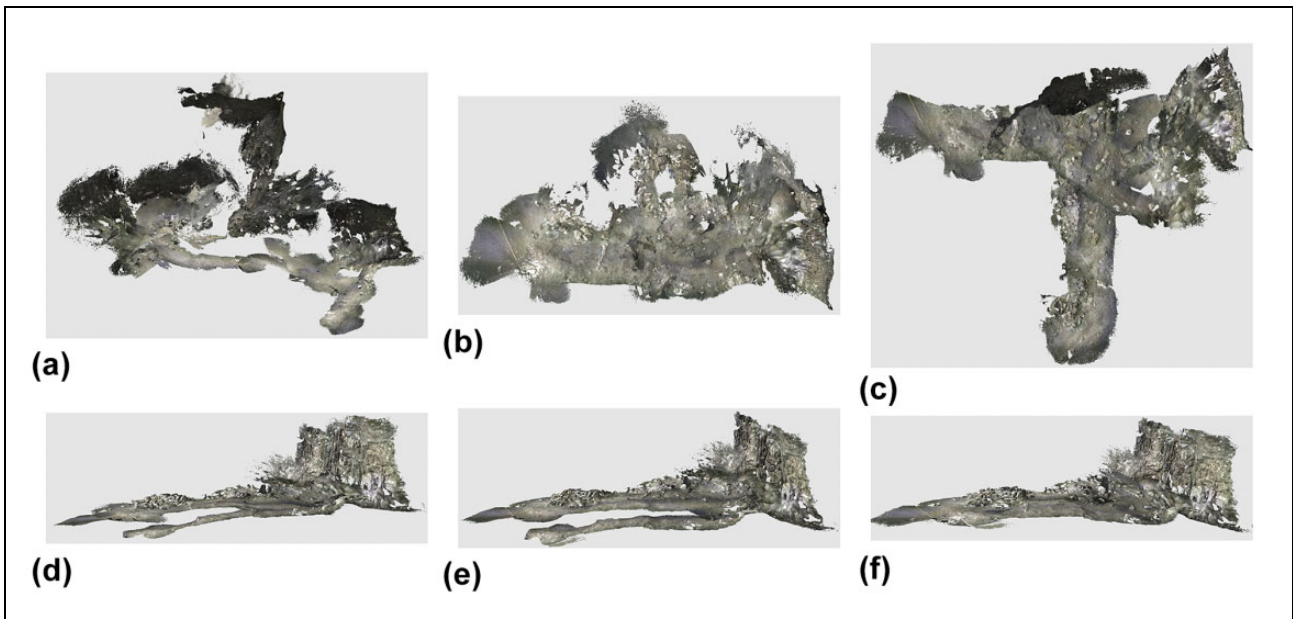


**Figure 1.** A part of a cave.

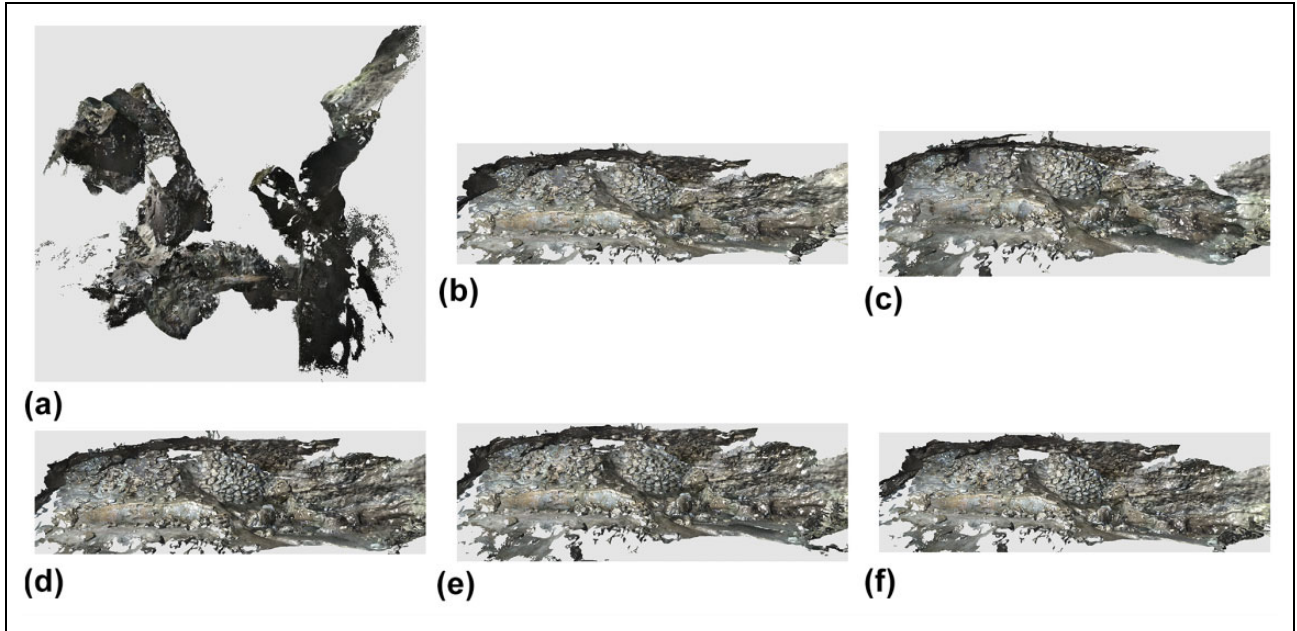
used to light up the environment as in Figure 1. In this data set, caves are scanned with a loopy motion for more loop optimization.

**Experiment design and baseline methods.** We evaluate modules of both tracking and loop closure from different approaches on the data we collected. For RGB-D data, there are two major different camera tracking methods, which are sparse feature-based and dense frame-to-model approaches. We choose the ORB-SLAM2 as the implementation for the sparse feature tracking and the ElasticFusion for the dense frame-to-model implementation. In experiments for tracking, both implementations have their loop closure disabled so that the difference can reflect tracking performance. In tracking of ElasticFusion, there is one important parameter that controls the weight of RGB in tracking. We use the default 10 for RGB-D tracking, and a number greater than 100 can disable RGB completely so that the tracking is totally based on the depth. For loop closure, there are three different types. ElasticFusion is using Ferns-ICP-based approach. ORB-SLAM2 utilized BoW. Our point cloud registration-based surface loop detection approach is the third type compared.

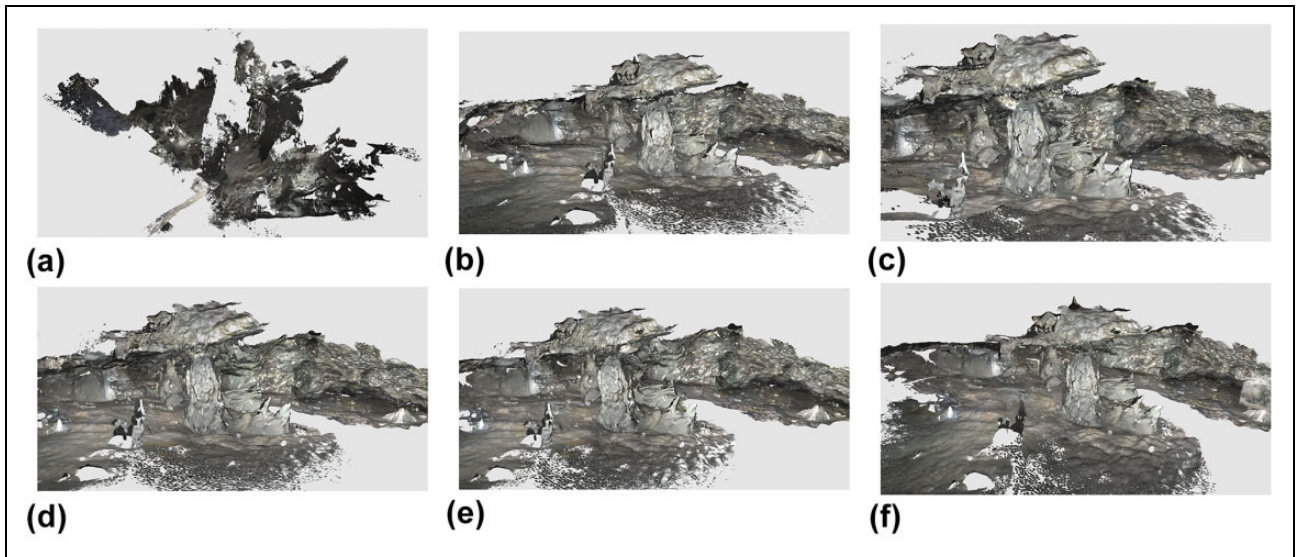
**Observations on tracking.** From Figures 2 to 5, we can see that RGB-D tracking of ElasticFusion does not work well on cave data, especially compared with when it uses depth only. We think this is due to the moving light source. RGB-D tracking calculates a transformation matrix partially by minimizing the intensity difference of two aligned images, which assumes the lighting condition of scenes is static.



**Figure 2.** Results on the chamber-floor-walking sequence. (a) ElasticFusion RGB-D tracking, (b) ElasticFusion Depth tracking, (c) ElasticFusion full, (d) ORB-SLAM2 tracking, (e) ORB-SLAM2 full, and (f) Ours. RGB-D: red, green, and blue-depth; ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.



**Figure 3.** Results on the chamber-entrance data sequence. (a) ElasticFusion RGB-D tracking, (b) ElasticFusion Depth tracking, (c) ElasticFusion full, (d) ORB-SLAM2 tracking, (e) ORB-SLAM2 full, and (f) Ours. RGB-D: red, green, and blue-depth; ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.

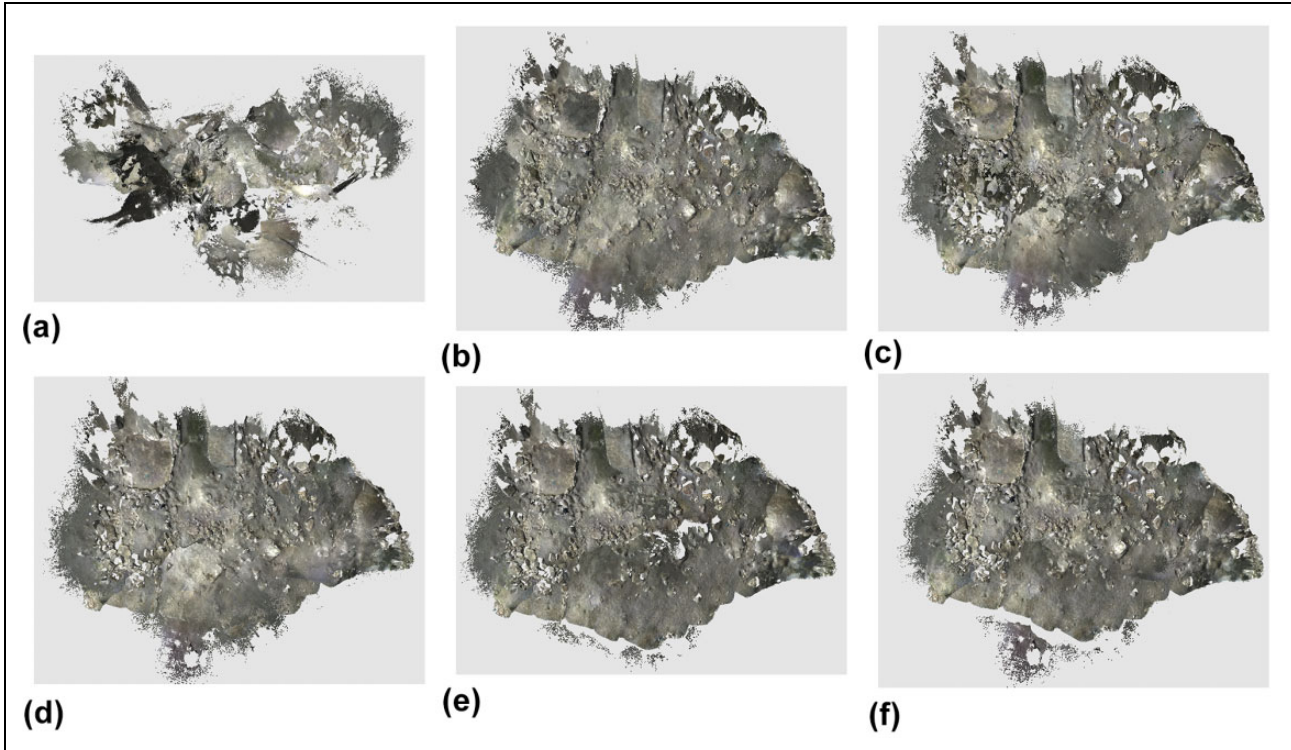


**Figure 4.** Results on the chamber-alcove data sequence. (a) ElasticFusion RGB-D tracking, (b) ElasticFusion Depth tracking, (c) ElasticFusion full, (d) ORB-SLAM2 tracking, (e) ORB-SLAM2 full, and (f) Ours. RGB-D: red, green, and blue-depth; ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.

The ElasticFusion depth tracking working quite well in most sequences excepts on the chamber-floor-walking data sequence. We can see in Figure 2(b) that half of the floor data get rotated around  $90^\circ$  clockwise. It is almost impossible for a dense direct tracking to recover from the error; due to that, there are no strong correspondences. The tracking of ORB-SLAM2 performed very well in all the data sequences, and it provides the possibility of globally

optimize the map. The robustness of feature-based tracking implemented by ORB-SLAM2 is the reason that we use it as our tracking module.

*Observations on loop detection and optimization.* When we compare the performance of loop closure, our surface-focused method performs the best. It connects important loops in all four sequences. The difficult data are the



**Figure 5.** Results on the chamber-cave-floor data sequence. (a) ElasticFusion RGB-D tracking, (b) ElasticFusion Depth tracking, (c) ElasticFusion full, (d) ORB-SLAM2 tracking, (e) ORB-SLAM2 full, and (f) Ours. RGB-D: red, green, and blue-depth; ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.

**Table 3.** Mean surface reconstruction error (in m) on augmented ICL-NUIM sequences.

	Living room 1	Living room 2	Office 1	Office 2	Average
CZK	0.033	0.028	0.019	0.022	0.026
Tracking	0.031	0.022	0.019	0.014	0.022
ORB-SLAM2	0.017	0.010	0.015	0.013	0.014
Ours	<b>0.007</b>	<b>0.007</b>	<b>0.013</b>	<b>0.010</b>	<b>0.009</b>

ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.

The best performance is marked in bold.

chamber-floor-walking one, as shown in Figure 2. Neither ElasticFusion nor ORB-SLAM2 tracks the camera trajectory correctly. Even after their loop closure, mismatches are still significant. Our method shows its robustness by reconstructing consistent 3D models on all data.

### Augmented ICL-NUIM data set

We use the augmented ICL-NUIM data set<sup>35</sup> to quantitatively analyze the performance of our system. This data set is a synthetic data set with ground truth surface models and camera trajectories. It has two indoor scenes: a living room and an office, and four RGB-D sequences, two sequences for each scene.

**Experiment design.** Evaluation metrics are camera trajectory translation root mean square error (RMSE) described by Handa et al. and the mean distance of the reconstructed surfaces to the ground truth surfaces in the same way as Whelan et al. We report them separately in Tables 3 and 4. Since different systems use different ways to fuse 3D models, for a fair comparison, we fuse 3D models using ElasticFusion using the same parameters with a camera trajectory estimate from each system. We use truncating depth distance of 4 m and 10 as the surfel confidence threshold for fusion.

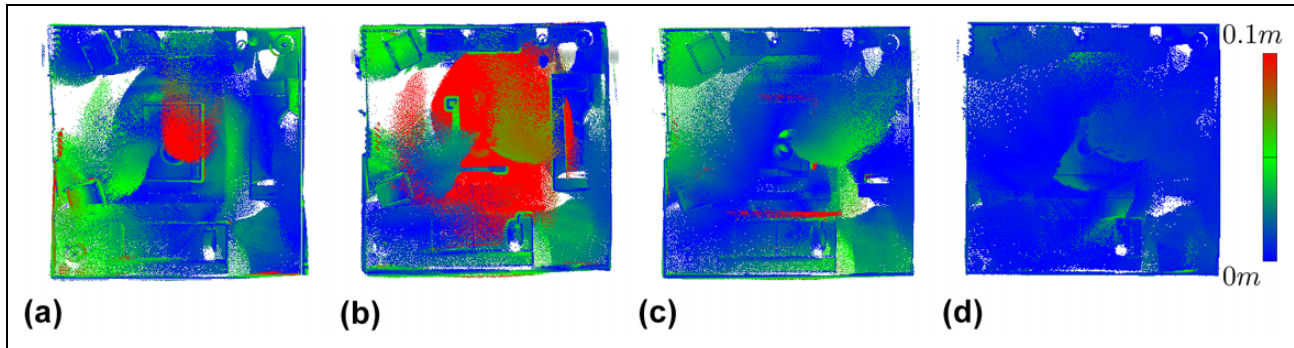
**Observations.** From Tables 3 and 4, our system can give best results on all data sequence in terms of both trajectory and surface estimation accuracy. To give a more informative

**Table 4.** RMSE (in meters) of estimated camera trajectories.

	Living room 1	Living room 2	Office 1	Office 2	Average
CZK	0.10	0.13	0.06	0.07	0.09
Tracking	0.14	0.05	0.05	0.03	0.07
ORB-SLAM2	0.10	0.03	0.04	0.03	0.05
Ours	<b>0.03</b>	<b>0.02</b>	<b>0.03</b>	<b>0.02</b>	<b>0.03</b>

RMSE: root mean square error; ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.

The best performance is marked in bold.



**Figure 6.** Distance error map of reconstructed models from different methods against ground truth on living room 1 data sequence. (a) CZK, (b) tracking, (c) ORB-SLAM2, and (d) Ours. ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.

comparison, we report an error map of the reconstructed model in Figure 6 on Living room 1 data sequence. We can see our results that have the lowest error across the whole model. Our method performs better because more loops get detected for improving the final loop optimization results. In ORB-SLAM2, the loop detector has trouble detecting some important loops because its consistency check can hardly be satisfied due to very local view overlappings. On the other hand, our surface loop detector does not have this problem. Compared to CZK, the performance gain comes from the advantage of sparse feature-based BA optimization, which can produce MAP results, thus more accurate camera trajectories and better reconstructed 3D models.

### SUN3D data set

The SUN3D data set<sup>40</sup> is a large-scale RGB-D database that captures many places. It contains many data sequences. And there are eight sequences (<http://sun3d.cs.princeton.edu/listNow.html>) that are labeled with object annotations and widely used for evaluating SLAM and 3D reconstruction systems. Since there is no ground truth available, we follow this practice and run experiments on these sequences. We show qualitative results in the form of screenshots of reconstructed 3D models in Figure 7.

For the sequences in Figure 7, we highlight the mismatches in tracking results so that readers can better compare them with our results. For sequences harvard\_c5, harvard\_c6, and harvard\_c8, there are no loops detected

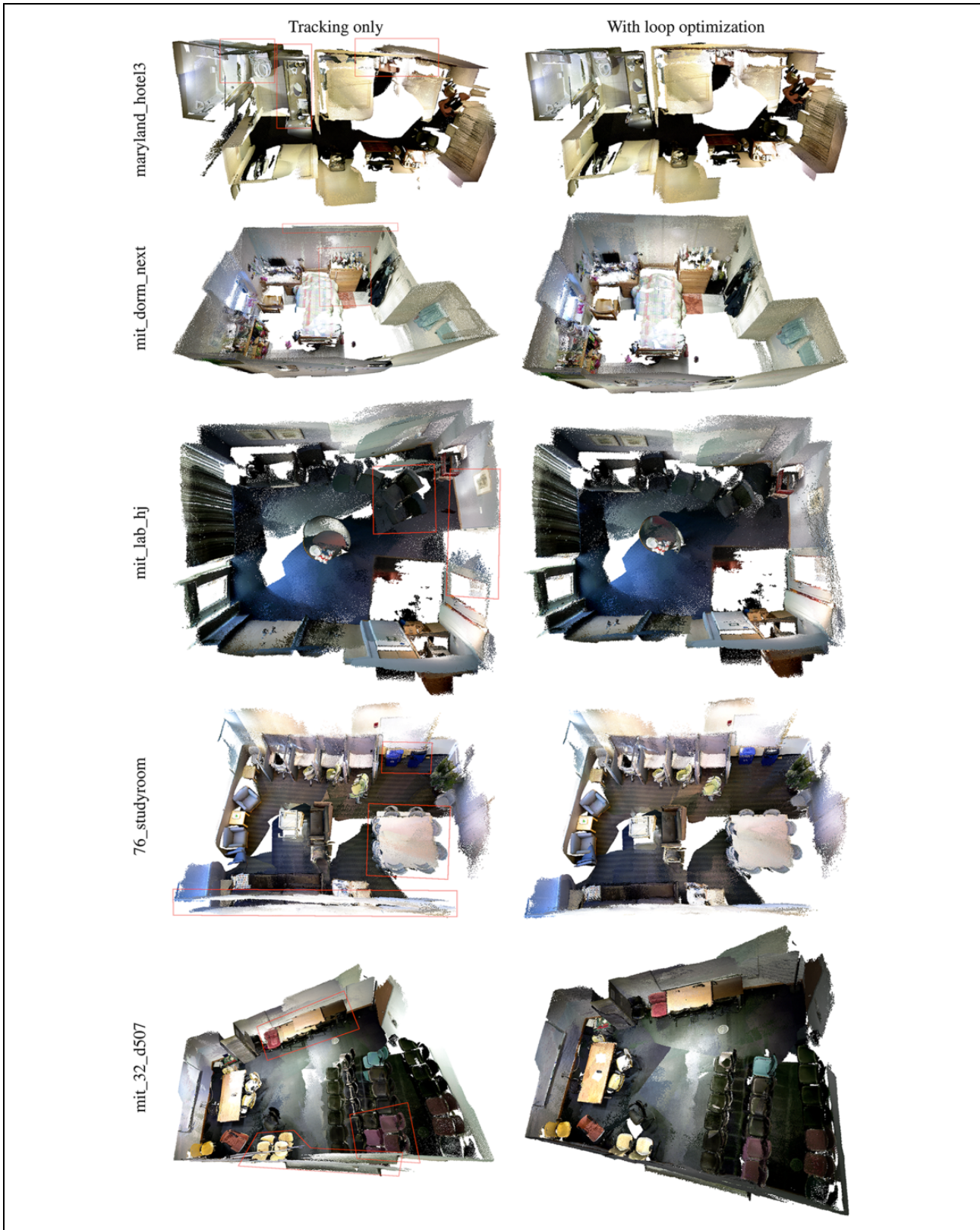
on top of tracking. So we do not include screenshots for them.

*Observations.* SUN3D data sequences are scanned with very loopy motion in some areas but only once for some other scene parts, thus is considered more difficult. Even though our method does not produce perfect reconstructed 3D models, it dramatically removed some significant mismatches. Also, our results are on par or better than other methods. Interested readers may compare with the results of CZK on this webpage (<http://redwood-data.org/indoor/models.html>).

### TUM RGB-D data set

The TUM data set<sup>39</sup> is an RGB-D data set that is commonly used to evaluate SLAM systems. This data set has 39 RGB-D sequences recorded in office and industrial environments with a large variety of camera motions and scenes. Along with RGB-D sequences, ground truth camera trajectories that are recorded with a motion capture system are also available. Following common practice, we run experiments only on sequences that are commonly used for SLAM evaluation.<sup>7</sup>

*Observations.* From the results listed in Table 5, it shows that for data sequence fr1/desk, fr1/room, fr2/desk, and fr3/office, our method makes improvement on tracking and achieves competitive results comparing full ORB-



**Figure 7.** Results on sequences of the SUN3D data set. The left column shows the results of the tracking part. The right column shows the results after our loop optimization. We highlight the mismatches in tracking results so that differences are easier compared.

SLAM2. Among these sequences, fr1/room has high absolute trajectory error (ATE) in tracking, and our surface loop significantly reduced the error. For the small performance difference, we think that it is due to the implementation difference of the tracking part and that the difference only has only marginal effects on reconstructed 3D models when ATE error is less than 0.02 as presented in Table 4 and Figure 6. For sequences fr1/desk2 and fr2/xyz, the tracking has provided good enough results that most of the frames are connected in the covisibility graph such that there are no loops to be detected. The sequence fr3/nst is a scan of a flat wall with rich texture but no geometry changes, so our method cannot detect surface loops in it; thus, no improvement is made. The TUM data set indicates that even though our method is designed for larger-scale environments with rich geometry changes, it can produce competitive results for some small-scale environments.

### Other public real-world scenes

We also run experiments on public real-world data sequences for qualitative analysis and a robustness test. We run our system on Copyroom and Lounge data sequences from Zhou et al. and DysonLab data set from

**Table 5.** TUM RGB-D data set comparison ATE (m).

Sequence name	ORB-SLAM2	Tracking	Ours
fr1/desk	0.016	0.021	0.019
fr1/desk2	0.022	0.028	0.028
fr1/room	0.047	0.295	0.068
fr2/desk	0.009	0.016	0.015
fr2/xyz	0.004	0.011	0.011
fr3/office	0.010	0.025	0.013
fr3/nst	0.019	0.034	0.034

RGB-D: red, green, and blue-depth; ATE: absolute trajectory error; ORB: oriented FAST and rotated BRIEF; SLAM: simultaneous localization and mapping; FAST: features from accelerated segment test; BRIEF: binary robust independent elementary features.

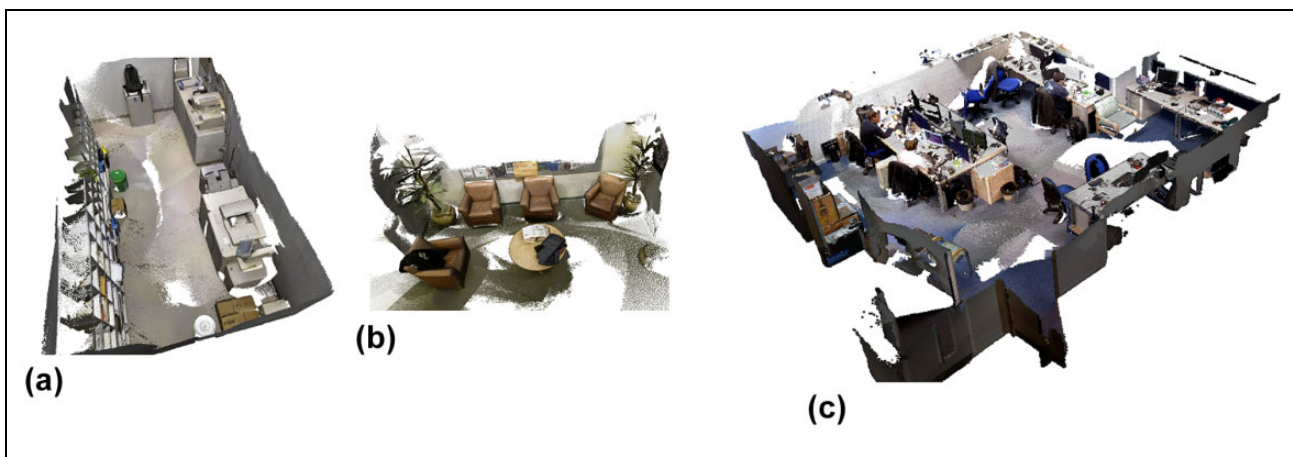
Whelan et al. We report our qualitative results in Figure 8. Since there are no ground truth models available, we only report screenshots of reconstructed models from our system. Due to space limitations, we do not include the results of other systems. Interested readers can refer to the authors' release. Visually, our system produces results at least matching the state-of-the-art methods.

### Conclusion

This article presents a novel 3D reconstruction system that maps both large archeological caves and general indoor environments with RGB-D cameras. The proposed system produces accurate 3D models by detecting and optimizing surface loops in sparse feature-based visual SLAM systems. By adding surface loop closure into a vSLAM system, globally consistent and optimal 3D models are generated accurately. The proposed system consists of five components: (1) sparse feature-based camera tracking from ORB-SLAM2, (2) surface model fusion powered by Surfels, (3) a novel fast surface loop detection algorithm, (4) a novel surface loop filtering method, and (5) loop optimization based on sparse feature-based BA.

For fast surface loop detection, we propose and implement a CUDA-based global point cloud registration algorithm that is parallelized to run on GPUs for faster RANSAC hypotheses testing. The proposed point cloud registration algorithm can finish in around 20 ms on NVIDIA Titan X Pascal without any precision and recall performance losses. The accelerated computational speed makes it possible to be used as a loop detection method.

In the surface loop filtering component part, a novel objective function is proposed to remove false-positive loops from entering the loop optimization step. The proposed objective function formulates a least-square pose graph with a BA term as the supporting backbone graph and robust least square terms with switchable constraints for surface loop detections. Due to the inherent difference in underlying optimization pose graphs, compared with its



**Figure 8.** Reconstructed models of real-world scenes. (a) Copyroom, (b) Lounge, and (c) DysonLab.

closely related work: CZK, our method is less strict to use in practical scenarios, especially on a fast-moving UAV platform. Because our method does not require maintaining RGB-D cameras facing surfaces all the time, but CZK requires it. In addition to the flexibility, the proposed method is benchmarked against CZK on the augmented ICL-NUIM data set in terms of filtered loop detection precision and recall. Experiments show that the proposed method performs better (with +37.5% precision and equally better recall) than CZK when only a limited number of loops are detected and provides competitive performance on all other scenarios.

Moreover, experiments are conducted to evaluate the performance of the full novel system on multiple data sets, including our Maya cave data set, augmented ICL-NUIM, TUM RGB-D data set, SUN3D data set, and some other public data sequences. The results are evaluated with ATE or trajectory RMSE when ground truth camera trajectories are available, surface reconstruction error when ground truth 3D models are accessible, and visual comparisons when no ground truth is available. The results show that sparse feature-based camera tracking performs the best in cave environments. The results also show that the proposed system produces the most reliable and accurate 3D reconstruction performance when surface loops are detected, filtered, and optimized on a sparse feature-based objective function. Other than in cave environments, experiments on other data sets show that the proposed system produces results on-par or better than baseline methods.


### Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by CITRIS Seed Grant entitled “Drones for Cave Archeology and 3D Mapping”.

### ORCID iD

YangQuan Chen  <https://orcid.org/0000-0002-7422-5988>

### References

- Zhang G, Shang B, Chen Y, et al. SmartCaveDrone: 3D cave mapping using UAVs as robotic co-archaeologists. In: *Proceeding of 2017 international conference on unmanned aircraft systems (ICUAS)*, Miami, FL, USA, 13–16 June 2017, IEEE. DOI: 10.1109/icuas.2017.7991499.
- Cadena C, Carlone L, Carrillo H, et al. Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. *IEEE Trans Robot* 2016; 32(6): 1309–1332.
- Zhang G, Chen Y, and Moyes H. Optimal 3D reconstruction of caves using small unmanned aerial systems and RGB-D cameras. In: *Proceeding of the 2018 international conference on unmanned aircraft systems (ICUAS)*, Dallas, TX, USA, 12–15 June 2018, IEEE, pp. 410–415. DOI: 10.1109/icuas.2018.8453277.
- Lefloch D, Kluge M, Sarbolandi H, et al. Comprehensive use of curvature for robust and accurate online surface reconstruction. *IEEE Trans Pattern Anal Mach Intell* 2017; 39(12): 2349–2365.
- Endres F, Hess J, Sturm J, et al. 3-D Mapping with an RGB-D camera. *IEEE Trans Robot* 2014; 30(1): 177–187.
- Whelan T, Kaess M, Fallon MF, et al. Kintinuous: spatially extended KinectFusion. In: *Proceeding of the RSS workshop on RGB-D: advanced reasoning with depth cameras*, Sydney, Australia, 9–13 July, 2012.
- Mur-Artal R and Tardos JD. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans Robot* 2017; 33(5): 1255–1262.
- Galvez-López D and Tardos JD. Bags of binary words for fast place recognition in image sequences. *IEEE Trans Robot* 2012; 28(5): 1188–1197.
- Newcombe RA, Izadi S, Hilliges O, et al. KinectFusion: real-time dense surface mapping and tracking. In: *Proceeding of the 2011 10th IEEE international symposium on mixed and augmented reality*, Basel, Switzerland, 26–29 October 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.
- Whelan T, Leutenegger S, Moreno RS, et al. ElasticFusion: dense SLAM without a pose graph. In: *Proceeding of the robotics: science and systems XI*. Robotics: science and systems foundation. DOI: 10.15607/rss.2015.xi.001.
- Davison AJ and Kita N. 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In: *Proceeding of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, Kauai, HI, USA, 8–14 December 2001, volume 1, pp. 384–391. DOI: 10.1109/CVPR.2001.990501.
- Montemerlo M and Thrun S. Simultaneous localization and mapping with unknown data association using FastSLAM. In: *Proceeding of the 2003 IEEE international conference on robotics and automation (Cat. No.03CH37422)*, Taipei, 14–19 September 2003, volume 2, pp. 1985–1991. DOI: 10.1109/ROBOT.2003.1241885.
- Castle RO, Gawley DJ, Klein G, et al. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In: *Proceeding of 2007 IEEE international conference on robotics and automation*, Rome, Italy, 10–14 April 2007, pp. 4102–4107. DOI: 10.1109/ROBOT.2007.364109.
- Dai A, Nießner M, Zollhöfer M, et al. BundleFusion: real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans Graph* 2017; 36(3): 1–18.
- Cummins M and Newman P. Probabilistic appearance based navigation and loop closing. In: *Proceeding of the 2007 IEEE international conference on robotics and automation*, Rome, Italy, 10–14 April 2007, pp. 2042–2048. DOI: 10.1109/ROBOT.2007.363622.

16. Angeli A, Filliat D, Doncieux S, et al. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Trans Robot* 2008; 24(5): 1027–1037.
17. Angeli A, Doncieux S, Meyer JA, et al. Real-time visual loop-closure detection. In: *Proceeding of the 2008 IEEE international conference on robotics and automation*, Pasadena, CA, USA, 19–23 May 2008, IEEE, pp. 1842–1847. DOI: 10.1109/ROBOT.2008.4543475.
18. Williams B, Klein G, and Reid I. Automatic relocalization and loop closing for real-time monocular SLAM. *IEEE Trans Pattern Anal Mach Intell* 2011; 33(9): 1699–1712.
19. Nicosevici T and Garcia R. Automatic visual bag-of-words for online robot navigation and mapping. *IEEE Trans Robot* 2012; 28(4): 886–898.
20. Yue H and Chen W. Comments on automatic visual bag-of-words for online robot navigation and mapping. *IEEE Trans Robot* 2015; 31(1): 223–224.
21. Ciarfuglia TA, Costante G, Valigi P, et al. A discriminative approach for appearance based loop closing. In: *Proceeding of the IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura-Algarve, Portugal, 7–12 October 2012, pp. 3837–3843. DOI: 10.1109/IROS.2012.6385654.
22. Bampis L, Amanatiadis A, and Gasteratos A. Fast loop-closure detection using visual-word-vectors from image sequences. *Int J Robot Res* 2018; 37(1): 62–82.
23. Han F, Wang H, Huang G, et al. Sequence-based sparse optimization methods for long-term loop closure detection in visual SLAM. *Auton Robot* 2018; 42(7): 1323–1335.
24. Bampis L, Amanatiadis A, and Gasteratos A. Encoding the description of image sequences: a two-layered pipeline for loop closure detection. In: *Proceeding of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Daejeon, Korea (South), 9–14 October 2016, pp. 4530–4536. DOI: 10.1109/IROS.2016.7759667.
25. Oh JH, Jeon JD, and Lee BH. Place recognition for visual loop-closures using similarities of object graphs. *Electron Lett* 2015; 51(1): 44–46.
26. Han F, Yang X, Deng Y, et al. SRAL: shared representative appearance learning for long-term visual place recognition. *IEEE Robot Autom Lett* 2017; 2(2): 1172–1179.
27. Lee GH and Pollefeys M. Unsupervised learning of threshold for geometric verification in visual-based loop-closure. In: *Proceeding of the IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, 31 May–7 June 2014, pp. 1510–1516. DOI: 10.1109/ICRA.2014.6907052.
28. Zhang G, Lilly MJ, and Vela PA. Learning binary features online from motion dynamics for incremental loop-closure detection and place recognition. In: *Proceeding of the 2016 IEEE international conference on robotics and automation (ICRA)*, Stockholm, Sweden, 16–21 May 2016, pp. 765–772. DOI: 10.1109/ICRA.2016.7487205.
29. Wu J, Zhang H, and Guan Y. An efficient visual loop closure detection method in a map of 20 million key locations. In: *Proceeding of the 2014 IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, 31 May–7 June 2014, pp. 861–866. DOI: 10.1109/ICRA.2014.6906955.
30. Xia Y, Li J, Qi L, et al. Loop closure detection for visual SLAM using PCANet features. In: *Proceeding of the international joint conference on neural networks (IJCNN)*, Vancouver, BC, Canada, 24–29 July 2016, pp. 2274–2281. DOI: 10.1109/IJCNN.2016.7727481.
31. Qin H, Huang M, Cao J, et al. Loop closure detection in SLAM by combining visual CNN features and submaps. In: *Proceeding of the automation and robotics (ICCAR) 2018 4th international conference on control*, Auckland, New Zealand, 20–23 April 2018, pp. 426–430. DOI: 10.1109/ICCAR.2018.8384713.
32. Cascianelli S, Costante G, Bellocchio E, et al. Robust visual semi-semantic loop closure detection by a covisibility graph and CNN features. *Robot Auton Syst* 2017; 92: 53–65.
33. Liu H, Zhao C, Huang W, et al. An end-to-end siamese convolutional neural network for loop closure detection in visual SLAM system. In: *Proceeding of the 2018 IEEE international conference on acoustics, speech and signal processing*, Calgary, AB, Canada, 15–20 April 2018.
34. Keller M, Lefloch D, Lambers M, et al. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In: *Proceeding of the 2013 international conference on 3d vision*, Seattle, WA, USA, 29 June–1 July 2013, IEEE. DOI: 10.1109/3dv.2013.9.
35. Choi S, Zhou QY, and Koltun V. Robust reconstruction of indoor scenes. In: *Proceeding of the 2015 IEEE conference on computer vision and pattern recognition (CVPR)*, Boston, MA, USA, 7–12 June 2015, IEEE. DOI: 10.1109/cvpr.2015.7299195.
36. Rusu RB, Blodow N, and Beetz M. Fast point feature histograms (FPFH) for 3D registration. In: *Proceeding of the 2009 IEEE international conference on robotics and automation*, Kobe, Japan, 12–17 May 2009, IEEE. DOI: 10.1109/robot.2009.5152473.
37. Fischler MA and Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 1981; 24(6): 381–395.
38. Sunderhauf N and Protzel P. Switchable constraints for robust pose graph SLAM. In: *Proceeding of 2012 IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura-Algarve, Portugal, 7–12 October 2012, IEEE. DOI: 10.1109/iros.2012.6385590.
39. Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems. In: *Proceeding of the 2012 IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura-Algarve, Portugal, 7–12 October 2012, IEEE. DOI: 10.1109/iros.2012.6385773.
40. Xiao J, Owens A, and Torralba A. SUN3D: a database of big spaces reconstructed using sfm and object labels. In: *Proceeding of the 2013 IEEE international conference on computer vision*, Sydney, NSW, Australia, 1–8 December 2013, IEEE. DOI: 10.1109/iccv.2013.458.