Computational Visual Media
DOI 10.1007/s41095-016-0072-2

Research Article

Feature-based RGB-D camera pose optimization for real-time 3D reconstruction

Chao Wang[1], **Xiaohu Guo**[1] (✉)

Abstract In this paper we present a novel feature-based RGB-D camera pose optimization algorithm for real-time 3D reconstruction systems. During camera pose estimation, current methods in online systems suffer from fast-scanned RGB-D data, or generate inaccurate relative transformations between consecutive frames. Our approach improves current methods by utilizing matched features across all frames and is robust for RGB-D data with large shifts in consecutive frames. We directly estimate camera pose for each frame by efficiently solving a quadratic minimization problem to maximize the consistency of 3D points in global space across frames corresponding to matched feature points. We have implemented our method within two state-of-the-art online 3D reconstruction platforms. Experimental results testify that our method is efficient and reliable in estimating camera poses for RGB-D data with large shifts.

Keywords camera pose optimization; feature matching; real-time 3D reconstruction; feature correspondence

1 Introduction

Real-time 3D scanning and reconstruction techniques have been applied to many areas in recent years with the prevalence of inexpensive depth cameras for consumers. The sale of millions of such devices makes it desirable for users to scan and reconstruct dense models of the surrounding environment by themselves. Online reconstruction techniques have various popular applications, e.g., in augmented reality (AR) to fuse supplemented elements with the real-world environment, in virtual reality (VR) to provide users with reliable environment perception and feedback, and in simultaneous localization and mapping (SLAM) for robots to automatically navigate in complex environments [1–3].

One of the earliest and most notable methods among RGB-D based online 3D reconstruction techniques is KinectFusion [4], which enables a user holding and moving a standard depth camera such as Microsoft Kinect to rapidly create detailed 3D reconstructions of a static scene. However, a major limitation of KinectFusion is that camera pose estimation is performed by frame-to-model registration using an iterative closest point (ICP) algorithm based on geometric data, which is only reliable for RGB-D data with small shifts between consecutive frames acquired by high-frame-rate depth cameras [4, 5].

To solve the aforementioned limitation, a common strategy adopted by most subsequent online reconstruction methods is to introduce photometric data into the ICP-based framework to estimate camera poses by maximizing the consistency of geometric information as well as color information between two adjacent frames [2, 5–11]. However, even though an ICP-based framework can effectively deal with RGB-D data with small shifts, it solves a non-linear minimization problem and always converges to a local minimum near the initial input because of the small angle assumption [4]. This indicates that pose estimation accuracy relies strongly on a good initial guess, which is unlikely to be satisfied if the camera moves rapidly or is shifted suddenly by the user. For the same reason, ICP-

1 University of Texas at Dallas, Richardson, Texas, USA. E-mail: C. Wang, chao.wang3@utdallas.edu; X. Guo, xguo@utdallas.edu (✉).

Manuscript received: 2016-09-09; accepted: 2016-12-20

based online reconstruction methods always generate results with drifts and distortion for scenes with large planar regions such as walls, ceilings, and floors, even if consecutive frames only contain small shifts. Figure 1 illustrates this shortcoming for several current online methods using an ICP-based framework, and also shows the advantage of our method on RGB-D data with large shifts on a planar region.

Another strategy to improve the robustness of camera tracking is to introduce RGB features into camera pose estimation by maximizing the 3D position consistency of corresponding feature points between frames [12–14]. These feature-based methods are better than ICP-based ones in handling RGB-D data with large shifts, since they simply run a quadratic minimization problem to directly compute the relative transformation between two consecutive frames [13, 14]. However, unlike ICP-based methods using frame-to-model registration, current feature-based methods estimate camera pose only based on pairs of consecutive frames, which usually brings in errors and accumulates drifts in reconstruction on RGB-D data with sudden change. Moreover, current feature-based methods always inaccurately estimate camera pose because of unreliable feature extractors and matching. Practically, the inaccurate camera poses are not utilized directly in reconstruction, but pushed into an offline backend post-process to improve their reliability, such as global pose graph optimization [12, 15] or bundle adjustment [13, 14]. For this reason, most current feature-based reconstruction methods are strictly offline.

In this paper, we combine the advantages of the two above strategies and propose a novel feature-based camera pose optimization algorithm for online 3D reconstruction systems. To solve the limitation that the ICP-based framework always converges to a local minimum near the initial input, our approach estimates the global camera poses directly by efficiently solving a quadratic minimization problem to maximize the consistency of matched feature points across frames, without any initial guess. This makes our method robust in dealing with RGB-D data with large shifts. Meanwhile, unlike current feature-based methods which only consider pairs of consecutive frames, our method utilizes matched features from all previous frames to reduce the impact of bad features and accumulated error in camera pose during scanning. This is achieved by keeping track of RGB features' 3D points information from all frames in a structure called the feature correspondence list.

Our algorithm can be directly integrated into current online reconstruction pipelines. We have implemented our method within two state-of-the-art online 3D reconstruction platforms. Experimental results testify that our approach is efficient and improves current methods in estimating camera pose on RGB-D data with large shifts.

2 Related work

Following KinectFusion, many variants and other brand new methods have been proposed to overcome its limitations and achieve more accurate reconstruction results. Here we mainly consider camera pose estimation methods in online

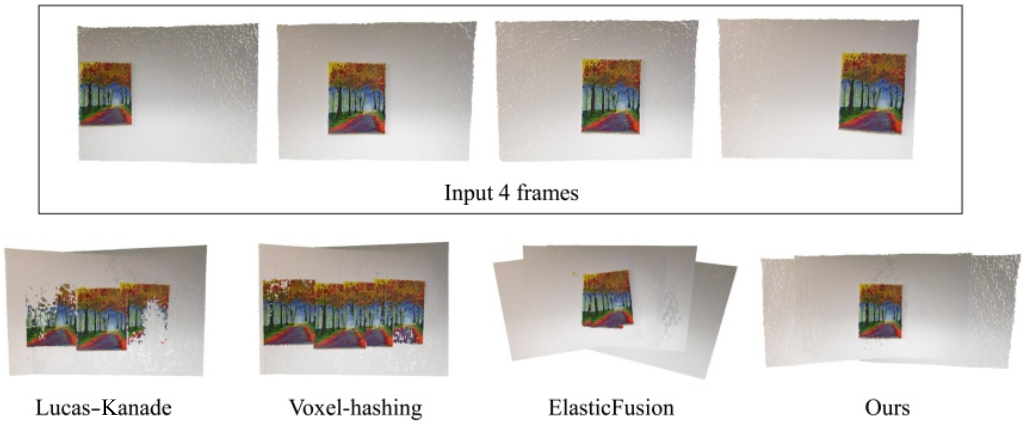


Fig. 1 Camera pose estimation comparison between methods. Top: four real input point clouds scanned using different views of a white wall with a painting. Bottom: results of stitching using camera poses provided by the Lucas–Kanade method [6], voxel-hashing [2], ElasticFusion [9], and our method.

and offline reconstruction techniques, and briefly introduce camera pose optimization in some other relevant areas.

2.1 Online RGB-D reconstruction

A typical online 3D reconstruction process takes RGB-D data as input and fuses the dense overlapping depth frames into one reconstructed model using some specific representation, of which two most important categories are volume-based fusion [2, 4, 5, 10, 16, 17] and point/surfel-based fusion [1, 9]. Volume-based methods are very common since they can directly generate models with connected surfaces, and are also efficient in data retrieval and use of the GPU. While KinectFusion is limited to a small fixed-size scene, several subsequent methods introduce different data processing techniques to extend the original volume structure, such as moving volume [16, 18], octree-based volume [17], patch volume [19], or hierarchical volume [20]. However, these online methods simply inherit the same ICP framework from KinectFusion to estimate camera pose.

In order to handle dense depth data and stitch frames in real time, most online reconstruction methods prefer an ICP-based framework which is efficient and reliable if the depth data has small shifts. While KinectFusion runs a frame-to-model ICP process with vertex correspondence obtained by projective data association, Peasley and Birchfield [6] improved it by providing ICP with a better initial guess and correspondence based on a warp transformation between consecutive RGB images. However, this warp transformation is only reliable for images with very small shifts, just like the ICP-based framework. Nießner et al. [2] introduced voxel-hashing technique into volumetric fusion to reconstruct scenes at large scale efficiently and used color-ICP to maintain geometric as well as color consistency of all corresponding vertices. Steinbrucker et al. [21] proposed an octree-based multi-resolution online reconstruction system which estimates relative camera poses between frames by stitching their photometric and geometric data together as closely as possible. Whelan et al.'s method [10] and a variant [5] both utilize a volume-shifting fusion technique to handle large-scale RGB-D data, while Whelan et al.'s ElasticFusion approach [9] extends it to a surfel-based fusion framework. They introduce local loop closure detection to adjust camera poses at any time during reconstruction. Nonetheless, these methods still rely on an ICP-based framework to determine a single joint pose constraint and therefore are still only reliable on RGB-D data with small shifts. Figure 1 gives a comparison between our method and these current methods on a rapidly scanned wall. In Section 4 we compare voxel-hashing [2], ElasticFusion [9], and our method on an RGB-D benchmark [22] and a real scene.

Feature-based online reconstruction methods are much rarer than ICP-based ones, since camera poses estimated only using features are usually unreliable due to the noisy RGB-D data, and must be subsequently post-processed. Huang et al. [13] proposed one of the earliest SLAM systems which estimates an initial camera pose in real time for each frame by utilizing FAST feature correspondence between consecutive frames, and sending all poses to a post-process for global bundle adjustment before reconstruction, which makes this method less efficient and not strictly an online reconstruction technique. Endres et al. [12] considered different feature extractors and estimated camera pose by simply computing the transformation between consecutive frames using an RANSAC algorithm based on feature correspondences. Xiao et al. [14] provided an RGB-D database with full 3D space views and used SIFT features to construct the transformation between consecutive frames, followed by bundle adjustment to globally improve pose estimates. In summary, current feature-based methods utilize feature correspondences only between pairs of consecutive frames to estimate the relative transformation between them. Unlike such methods, our method utilizes the feature-matching information from all previous frames by keeping track of the information in a feature correspondence list. Section 4.4 compares our method and current feature-based frameworks utilizing only pairs of consecutive frames.

2.2 Offline RGB-D reconstruction

The typical and most common scheme for offline reconstruction methods is to take advantage of some global optimization technique to determine consistent camera poses for all frames, such as bundle

adjustment [13, 14], pose graph optimization [5, 14, 23], and deformation graph optimization with loop closure detection [9]. Some offline works utilize similar strategies to online methods [2, 5, 9] by introducing feature correspondences into an ICP-based framework. They maximize the consistency of both dense geometric data and sparse image features, such as one of the first reconstruction systems proposed by Henry et al. [7] using SIFT features.

Other work introduces various special points of interest into camera pose estimation and RGB-D reconstruction. Zhou and Koltun [24] proposed an impressive offline 3D reconstruction method which focuses on preserving details of points of interest with high density values across RGB-D frames, and runs pose graph optimization to obtain globally consistent pose estimations for these points. Two other works by Zhou et al. [25] and Choi et al. [26] both detect smooth fragments as point of interest zones and attempt to maximize the consistency of corresponding points in fragments across frames using global optimization.

2.3 Camera pose optimization in other areas

Camera pose optimization is also very common in many other areas besides RGB-D reconstruction. Zhou and Koltun [3] presented a color mapping optimization algorithm for 3D reconstruction which optimizes camera poses by maximizing the color agreement of 3D points' 2D projections in all RGB images. Huang et al. [13] proposed an autonomous flight control and navigation method utilizing feature correspondence to estimate relative transformation between consecutive frames in real time. Steinbrücker et al. [27] presented a real-time visual odometry method which estimates camera poses by maximizing photo-consistency between consecutive images.

3 Camera pose estimation

Our camera pose optimization method attempts to maximize the consistency of matched features' corresponding 3D points in global space across frames. In this section we start with a brief overview of the algorithmic framework, and then describe the details of each step.

3.1 Overall scheme

The pipeline is illustrated in Fig. 2. For each input RGB-D frame, we extract the RGB features in the first step (see Section 3.2), and then generate a good feature match with correspondence-check (see Section 3.3). Next, we maintain and update a data structure called the feature correspondence list to store matched features and corresponding 3D points in the camera's local coordinate space across frames (see Section 3.4). Finally, we estimate camera pose by minimizing the difference between matched features' 3D positions in global space (see Section 3.5).

3.2 Feature extraction

2D feature points can be utilized to reduce the amount of data needed to evaluate the similarity between two RGB images while preserving the accuracy of the result. In order to estimate camera pose efficiently in real time while guaranteeing the reconstruction reliability, we need to select a feature extraction method with a good balance between feature accuracy and speed. We ignore corner-based feature detectors such as BRIEF and FAST, since the depth data from consumer depth cameras always contains much noise around object contours due to the cameras' working principles [28]. Instead, we simply use an SURF detector to extract and describe RGB features, for two main reasons. Firstly, SURF is robust, stable, and scale and rotation invariant [29], which is important for establishing reliable feature correspondences between images. Secondly, existing methods can efficiently compute SURF in parallel on the GPU [30].

3.3 Feature matching

Using the feature descriptors, a feature match can be obtained easily but it usually contains many

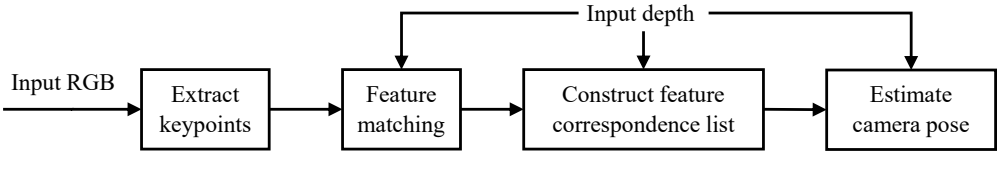


Fig. 2 Algorithm overview.

mismatched pairs. To remove as many outliers as possible, we run an RANSAC-based correspondence-check based on 2D homography and relative transformation between pairs of frames.

For two consecutive frames $i-1$ and i with RGB images and corresponding 3D points in the camera's local coordinate space, we first obtain an initial feature match between 2D features based on their descriptors. Next, we run a number of iterations, and in each iteration we randomly select 4 feature pairs to estimate the 2D homography $\boldsymbol{H}_z$ using the direct linear transformation algorithm [31] and the 3D relative transformation $\boldsymbol{T}_z$ between the corresponding 3D points. $\boldsymbol{H}_z$ and $\boldsymbol{T}_z$ with lowest re-projection errors amongst all feature pairs are selected as the final ones to determine the outliers. After iterations, feature pairs with a 2D re-projection error larger than a threshold σ_H or a 3D re-projection error larger than a threshold σ_T are treated as outliers, and are removed from the initial feature match.

During the correspondence-check, we only select feature pairs with valid depth values. Meanwhile, in order to reduce noise in the depth data, we pre-smooth the depth image with a bilateral filter before computing 3D points from 2D features. After the correspondence-check, if the number of valid matched features is too small, the estimated camera pose obtained based on them will be unreliable. Therefore, we abandon all subsequent steps after feature matching and use a traditional ICP-based framework if the number of validly matched features is smaller than a threshold σ_F. In our experiment, we empirically choose $\sigma_H = 3$, $\sigma_T = 0.05$, and $\sigma_F = 10$.

Figure 3 shows a feature matching comparison before and after the correspondence-check for two consecutive images captured by a fast-moving camera. The blue circles are feature points, while the green circles and lines are matched feature pairs. Note that almost all poorly matched correspondence pairs are removed.

3.4 Feature correspondence list construction

In order to estimate the camera pose by maximizing the consistency of the global positions of matched features in all frames, we establish and update a feature correspondence list (FCL) to keep track of matched features in both the spatial and temporal domain. The FCL is composed of 3D point sets, each of which denotes a series of 3D points in the camera's local coordinate space, whose corresponding 2D pixels are matched features across frames. Thus, the FCL in frame i is denoted by $\boldsymbol{L} = \{\boldsymbol{S}_j | j = 0, \ldots, m_i - 1\}$, where each $\boldsymbol{S}_j$ contains 3D points whose corresponding 2D points are matched features, j is the point set index, and m_i is the number of point sets in the FCL in frame i. The FCL can be simply constructed: Fig. 4 illustrates the process used to construct FCL for two consecutive frames.

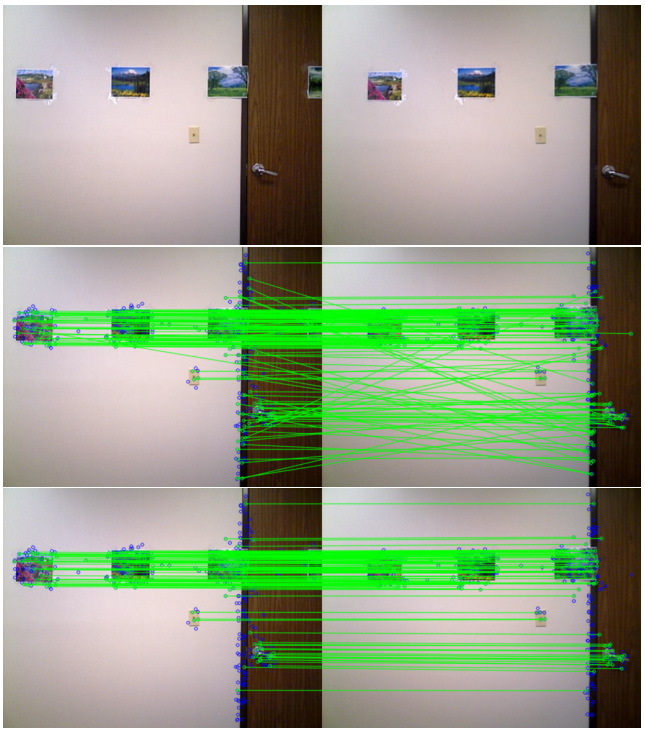


Fig. 3 Two original images (top), feature matching before (middle) and after (bottom) correspondence checking.

By keeping track of all RGB features' 3D positions in each camera's local space, we can estimate camera poses by maximizing the consistency of all these 3D points' global positions. By utilizing feature information from all frames instead of just two consecutive frames, we aim to reduce the impact of possible bad features, such as incorrectly matched features or features from ill-scanned RGB-D frames. Moreover, this also avoids the accumulation of error in camera pose from previous frames.

3.5 Camera pose optimization

For the 3D points in each point set in FCL, their

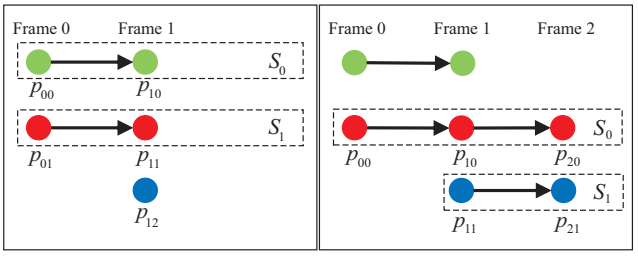


Fig. 4 Feature correspondence lists for frame 1 (left) and frame 2 (right). To construct the FCL for frame 2, we remove point sets with unmatched features (green), add matched points whose corresponding features are in the previous frame's FCL into the corresponding point sets (red), and add new point sets for matched features whose corresponding features are not in the previous frame's FCL (blue). Finally we re-index all points in the FCL. The number of point sets in the two FCLs is the same; here $m_1 = m_2 = 2$.

corresponding RGB features can be regarded as 2D projections from one 3D point in the real world on the RGB images in a continuous series of frames. For these 3D points in the camera coordinate space, we aim to ensure that their corresponding 3D points in the world space are as close as possible.

Given the FCL $\boldsymbol{L} = \{\boldsymbol{S}_j | j = 0, \cdots, m_i - 1\}$ in frame i, for each 3D point $\boldsymbol{p}_{ij} \in \boldsymbol{S}_j$, our objective is to maximize the agreement between $\boldsymbol{p}_{ij}$ and its target position in the world space with respect to a rigid transformation. Specifically, we seek a rotation $\boldsymbol{R}_i$ and translation vector $\boldsymbol{t}_i$ that minimize the following energy function:

$$E_i(\boldsymbol{R}_i, \boldsymbol{t}_i) = \sum_{j=0}^{m_i-1} w_j \|\boldsymbol{R}_i \boldsymbol{p}_{ij} + \boldsymbol{t}_i - \boldsymbol{q}_j\|^2 \quad (1)$$

where w_j is a weight to distinguish the importance of points, and $\boldsymbol{q}_j$ is the target position in the world space of $\boldsymbol{p}_{ij}$ after transformation. In our method we initially set:

$$\boldsymbol{q}_j = \frac{1}{|\boldsymbol{S}_j| - 1} \sum_{k=n_j}^{i-1} (\boldsymbol{R}_k \boldsymbol{p}_{kj} + \boldsymbol{t}_k) \quad (2)$$

which is the average position of the 3D points in the world frame obtained from all points in $\boldsymbol{S}_j$ except for $\boldsymbol{p}_{ij}$ itself, where n_j is the frame index for $\boldsymbol{S}_j$'s first point. Intuitively, the more frequently a 3D global point appears in frames, the more reliable this point's measured data will be for the estimation of camera pose. Therefore, we use $w_j = |\boldsymbol{S}_j|$ to balance the importance of points. $\boldsymbol{q}_j$ in Eq. (2) can be easily computed from the stored information in frame i's FCL.

The energy function $E_i(\boldsymbol{R}_i, \boldsymbol{t}_i)$ in Eq. (1) is a quadratic least-squares objective and can be minimized by Arun et al.'s method [32]:

$$\boldsymbol{R}_i = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}} \quad (3)$$

$$\boldsymbol{t}_i = \overline{\boldsymbol{q}} - \boldsymbol{R}_i \overline{\boldsymbol{p}}_i \quad (4)$$

Here $\boldsymbol{D} = \mathrm{diag}(1, 1, \det(\boldsymbol{V}\boldsymbol{U}^{\mathrm{T}}))$ ensures that $\boldsymbol{R}_i$ is a rotation matrix without reflection. $\boldsymbol{U}$, $\boldsymbol{V}$ are both $3 \times m_i$ matrices from the singular value decomposition (SVD) of matrix $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, which is constructed by $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{W}\boldsymbol{Y}^{\mathrm{T}}$ where:

$$\boldsymbol{X} = [\boldsymbol{p}_{i0} - \overline{\boldsymbol{p}}_i, \ldots, \boldsymbol{p}_{i(m_i-1)} - \overline{\boldsymbol{p}}_i] \quad (5)$$

$$\boldsymbol{W} = \mathrm{diag}(w_0, \ldots, w_{m_i-1}) \quad (6)$$

$$\boldsymbol{Y} = [\boldsymbol{q}_0 - \overline{\boldsymbol{q}}, \ldots, \boldsymbol{q}_{m_i-1} - \overline{\boldsymbol{q}}] \quad (7)$$

$$\overline{\boldsymbol{p}}_i = \frac{\sum_k w_k \boldsymbol{p}_{ik}}{\sum_k w_k}, \ \overline{\boldsymbol{q}} = \frac{\sum_k w_k \boldsymbol{q}_k}{\sum_k w_k} \quad (8)$$

Here $\boldsymbol{X}$ and $\boldsymbol{Y}$ are both $3 \times m_i$ matrices, $\boldsymbol{W}$ is a diagonal matrix with weight values, and $\overline{\boldsymbol{p}}_i$ and $\overline{\boldsymbol{q}}$ are the mass centers of all $\boldsymbol{p}_{ij}$ and $\boldsymbol{q}_j$ in frame i respectively. In general, by minimizing the energy function in Eq. (1), we seek a rigid transformation which makes each 3D point's global position in the world space as close as possible to the average position of all its corresponding 3D points from all previous frames.

After solving Eq. (1) for the current frame i, each $\boldsymbol{p}_{ij}$'s target position $\boldsymbol{q}_j$ in Eq. (2) can be updated by

$$\boldsymbol{q}'_j = \frac{1}{|\boldsymbol{S}_j|} \sum_{k=n_j}^{i} (\boldsymbol{R}_k \boldsymbol{p}_{kj} + \boldsymbol{t}_k) \quad (9)$$

This is simply done by putting $\boldsymbol{p}_{ij}$ and the newly obtained transformation $\boldsymbol{R}_i$ and $\boldsymbol{t}_i$ into Eq. (2), and estimating $\boldsymbol{q}_j$ as the average center of all points in $\boldsymbol{S}_j$. Note that we can utilize the new $\boldsymbol{q}'_j$ in Eq. (9) to further decrease the energy in Eq. (1) and obtain another new transformation, which can be utilized again to update $\boldsymbol{q}'_j$ in turn. Therefore, an iterative optimization process updating $\boldsymbol{q}'_j$ and minimizing the energy E_i can be repeatedly used to optimize the transformation until the energy converges.

Furthermore, the aforementioned iterative process can also be run on previous frames to further maximize the consistency of matched 3D points' global positions between frames. If an online reconstruction system contains techniques to update the previously reconstructed data, then the further optimized poses in previous frames can be used to update the reconstruction quality further. Actually, we only need to optimize poses between frame r to

i, where r is the earliest frame index of all points in frame i's FCL. A common case during online scanning and reconstruction is that, the camera stays steady on a same scene for a long time. Then, the correspondence list will keep too many old redundant matched features from very early previous frames, which will greatly increase the computation cost of optimization. To avoid this, we check the gap between r and i for every frame i. If $i-r$ is larger than a threshold δ, we only run optimization between frame $i-\delta$ and i. In the experiments, we use $\delta = 50$.

In particular, minimizing each energy $E_k (r \leqslant k \leqslant i)$ is equivalent to minimizing the sum of the energy between these frames:

$$E(E_r, \ldots, E_i) = \sum_{k=r}^{i} E_i = \sum_{k=r}^{i} \sum_{j=0}^{m_k-1} w_j \| \boldsymbol{R}_k \boldsymbol{p}_{kj} + \boldsymbol{t}_k - \boldsymbol{q}_j \|^2 \quad (10)$$

According to the solutions in Eqs. (5)–(8), the computation of each transformation $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$ in Eq. (10) is independent of that in other frames. The total energy E is estimated each time in the iterative optimization process to determine if the convergence condition is satisfied or not.

Algorithm 1 describes the entire iterative camera pose optimization process in our method. In the experiments we set the energy threshold $\varepsilon = 0.01$. Our optimization method is very efficient in that it only takes $O(m_i)$ multiplications and additions as well as a few SVD processes on 3×3 matrices.

Algorithm 1 Camera pose optimization

Input: Feature correspondence list for frame i, earliest frame index r, and energy threshold ε.

Output: Optimized camera poses between frame r and frame i.

```
Update {q_j} via Eq. (2);
Compute energy E_i in Eq. (1) with {q_j} and obtain R_i and t_i;
Compute total energy E in Eq. (10);
while (true) do
    E' ⇐ E;
    Update {q'_j} with {R_k, t_k | k = r, ..., i} via Eq. (9);
    for all (r ⩽ k ⩽ i) do
        Compute energy E_k in Eq. (1) with {q'_j} and obtain R'_k and t'_k ;
    end for
    Compute new energy E in Eq. (10) with {R'_k, t'_k | k = r, ..., i};
    if (|E' − E| < ε) then
        break;
    end if
    for all (r ⩽ k ⩽ i) do
        R_k ⇐ R'_k, t_k ⇐ t'_k;
    end for
end while
Return {R'_k, t'_k | k = r, ..., i}
```

4 Experimental results

To assess the capabilities of our camera pose estimation method, we embedded it within two state-of-the-art platforms: a volume-based method based on voxel-hashing [2] and a surfel-based method, ElasticFusion [9]. In the implementation, we first estimate camera poses using our method, and then regard them as good initial guesses for the original ICP-based framework in each platform. The reason is that the reconstruction quality is possibly low if the online system does not run a frame-to-model framework to stitch dense data from the current frame with the previous model during reconstruction [5]. Note that for each frame, even though our method optimizes camera poses from all relevant frames, we only use the optimized pose for the current frame for the frame-to-model framework to update the reconstruction, and the optimized poses in previous frames are only utilized to estimate the camera poses in future frames.

4.1 Trajectory estimation

We first compare our method with both voxel-hashing [2] and ElasticFusion [9], evaluating the trajectory estimation performance using several datasets from the RGB-D benchmark [22]. In order to compare with ElasticFusion [9], we utilize the same error metric as in their work, absolute trajectory root-mean-square error (ATE) which measures the root-mean-square of Euclidean distances between estimated camera poses and ground truth ones associated with timestamps [9, 22].

Table 1 shows the results from each method with and without our improvement. We denote the smallest error for each dataset in bold. Here "dif1" and "dif5" denote the frame difference used for each dataset during reconstruction. In other words, for "dif5", we only use the first frame of every 5 consecutive frames in each original dataset, and omit the other 4 intermediate frames

Table 1 Trajectory estimation comparison of methods using ATE metric

System	fr1/desk		fr1/floor		fr1/room		fr3/ntf	
	dif1	dif5	dif1	dif5	dif1	dif5	dif1	dif5
Voxel-hashing	1.10	0.74	1.01	0.70	0.61	1.08	1.32	1.30
Ours	0.32	0.32	**0.16**	**0.19**	0.34	0.61	0.18	0.08
ElasticFusion	**0.03**	0.30	0.41	0.54	0.38	0.48	**0.08**	0.15
Ours	0.04	**0.21**	0.17	0.22	**0.32**	**0.35**	**0.08**	**0.08**

in order to estimate the trajectories on RGB-D data with large shifts, while for "dif1" we just use the original dataset. Note that our results are different when embedded in the two platforms even for the same dataset. This is because, firstly, the two online platforms utilize different data processing and representation techniques, and different frame-to-model frameworks during reconstruction. Secondly, the voxel-hashing platform does not contain any optimization technique to modify previously constructed models and camera poses, while ElasticFusion utilizes both local and global loop closure detection in conjunction with global optimization techniques to optimize previous data and generate a globally consistent reconstruction [9]. Results in Table 1 show that our method improves upon the other two methods for estimating trajectories, especially on large planar regions such as fr1/floor and fr3/ntf which both contain floor with textures. Furthermore, our method also estimates trajectories better than the other methods when the shifts between the RGB-D frames are large.

4.2 Pose estimation

To estimate the pose estimation performance, we compared our methods with the same two methods on the same benchmark using relative pose error (RPE) [22], which measures the relative pose difference between each estimated camera pose and the corresponding ground truth. Table 2 gives the results, which show that our method can improve camera pose estimation on datasets with large shifts, even though our result is only on a par with the others on the original datasets with small shifts between consecutive frames.

4.3 Surface reconstruction

In order to compare the influence of computed camera poses on the final reconstructed models for our method and the others, we firstly compute camera poses by each method on its corresponding platform, and then use all the poses on the same voxel-hashing platform to generate reconstructed models. Here our method runs on the voxel-hashing platform. Figure 5 gives the reconstruction results for different methods on the fr1/floor dataset from the same benchmark, with frame difference 5. The figure shows that our method improves the reconstructed surface by producing good camera poses for the RGB-D data with large shifts.

To test our method on a fast-moving camera on

Table 2 Pose estimation comparison of methods using RPE metric

System	fr1/desk		fr1/floor		fr1/room		fr3/ntf	
	dif1	dif5	dif1	dif5	dif1	dif5	dif1	dif5
Voxel-hashing	1.57	1.16	1.25	0.98	1.15	1.49	1.60	1.62
Ours	0.91	0.94	0.80	0.80	0.84	1.14	1.36	1.42
ElasticFusion	**0.04**	0.41	**0.42**	0.58	**0.51**	0.63	**0.11**	**0.11**
Ours	0.05	**0.29**	0.43	**0.48**	0.54	**0.61**	0.12	**0.11**

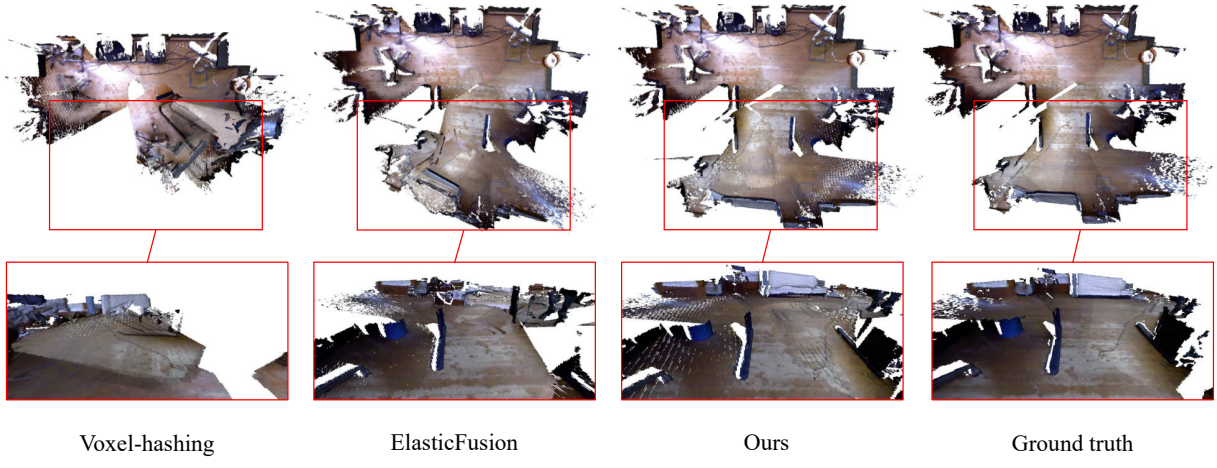


Fig. 5 Reconstruction results for different methods on fr1/floor from the RGB-D benchmark [22] with frame difference 5.

a real scene, we fixed an Asus XTion depth camera on a tripod with a motor to rotate the camera with controlled speed. With this device, we firstly scanned a room by rotating the camera only around its axis (the y-axis in the camera's local coordinate frame) for several rotations with a fixed speed, and selected the RGB-D data for exactly one rotation for the test. This dataset contains 235 RGB-D frames; most of the RGB images are blurred, since it took the camera only about 5 seconds to finish the rotation. Figure 6 gives an example showing two blurred images from this RGB-D dataset. Note that our feature matching method can still match features very well.

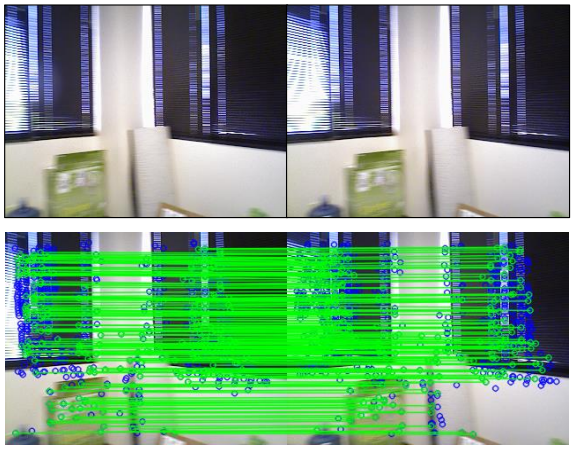


Fig. 6 Two blurred images (top) and feature matching result (bottom) from our scanned RGB-D data from a real scene using a fast-moving camera.

Figure 7 gives the reconstruction results produced by different methods on the dataset. As in Fig. 5, all reconstruction results here are also obtained using the voxel-hashing platform with camera poses pre-computed by different methods on each corresponding platform; again our method ran on the voxel-hashing platform. For the ground truth camera poses, since we scan the scene with fixed rotation speed, we simply compute the ground truth camera pose for each frame $i\,(0 \leqslant i < 235)$ as $\boldsymbol{R}_i = \boldsymbol{R}_y(\theta_i)$ with $\theta_i = (360(i-1)/235)^\circ$ and $\boldsymbol{t}_i = 0$, where $\boldsymbol{R}_y(\theta_i)$ rotates around the y-axis by an angle θ_i. Moreover, note that ElasticFusion [9] utilizes loop closure detection and deformation graph optimization to globally optimize camera poses and global point positions in the final model. To make the comparison more reasonable, we introduce the same loop closure detection in ElasticFusion [9] into our method, and use a pose graph optimization tool [15] to globally optimize camera poses for all frames efficiently. Figure 7 shows that our optimized camera poses can determine the structure of the reconstructed model very well for the real-scene data captured by a fast-moving camera.

4.4 Justification of feature correspondence list

In our method we utilize the FCL in order to

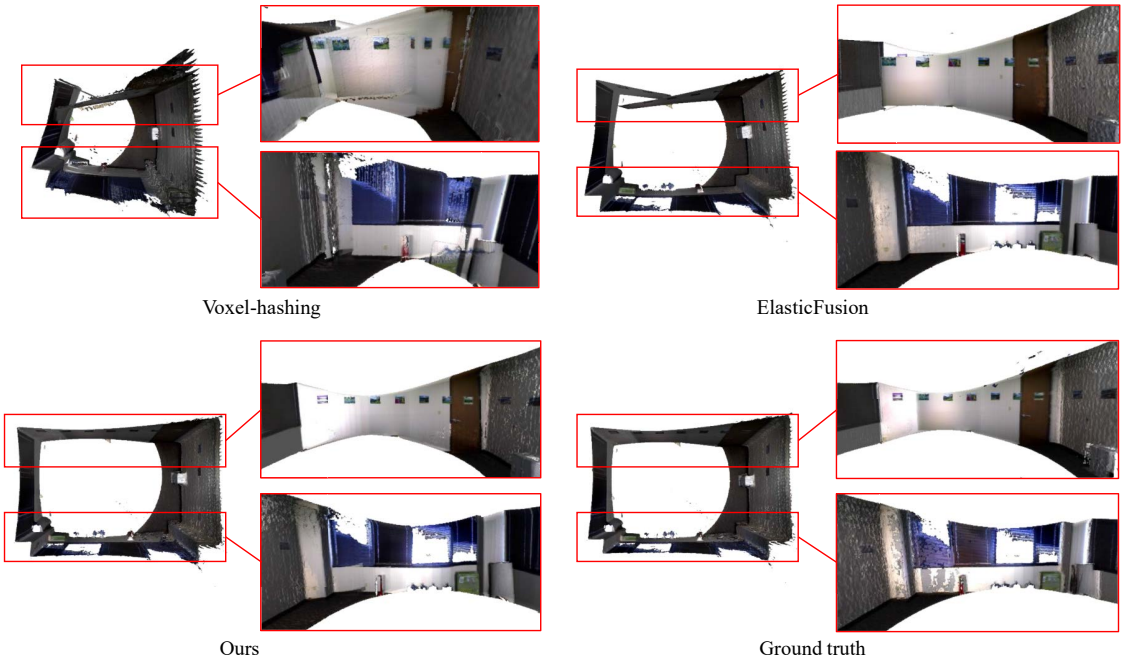


Fig. 7 Reconstruction results for different methods on room data captured by a speed-controlled fast-moving camera.

reduce the impact of bad features on camera pose estimation, and also to avoid accumulating error in camera poses during scanning. Current feature-based methods always estimate the relative transformation between the current frame and the previous one using only the matched features in these two consecutive frames [12–14] and here we call this strategy *consecutive-feature estimation*.

In our framework, the consecutive-feature estimation can be easily implemented by only using steps (1) and (2) (lines 1 and 2) in Algorithm 1 for each $\boldsymbol{q}_j = \boldsymbol{p}_{(i-1)j}$, which is $\boldsymbol{p}_{ij}$'s matched 3D point in the previous frame. Figure 8 gives the ATE and RPE errors for our method utilizing FCLs and the consecutive-feature method on fr1/floor, for increasing frame differences. Clearly our method with FCLs outperforms the consecutive-feature method in determining camera poses for RGB-D data with large shifts.

4.5 Performance

We have tested our method on the voxel-hashing platform on a laptop running Microsoft Windows 8.1 with an Intel Core i7-4710HQ CPU at 2.5 GHz, 12 GB RAM, and an NVIDIA GeForce GTX 860M GPU with 4 GB memory. We used the OpenSURF library and used OpenCL [30] to extract SURF features on each down-sampled 320 × 240 RGB image. For each frame, our camera pose optimization pipeline takes about 10 ms to extract features and finish feature matching, 1–2 ms for FCL construction, and only 5–8 ms for the camera pose optimization step, including the iterative optimization of camera poses for all relevant frames. Therefore, our method is efficient enough to run in real time. We also note that the offline pose graph optimization tool [15] used for the RGB-D data described in Section 4.3 takes only 10 ms for global pose optimization of all frames.

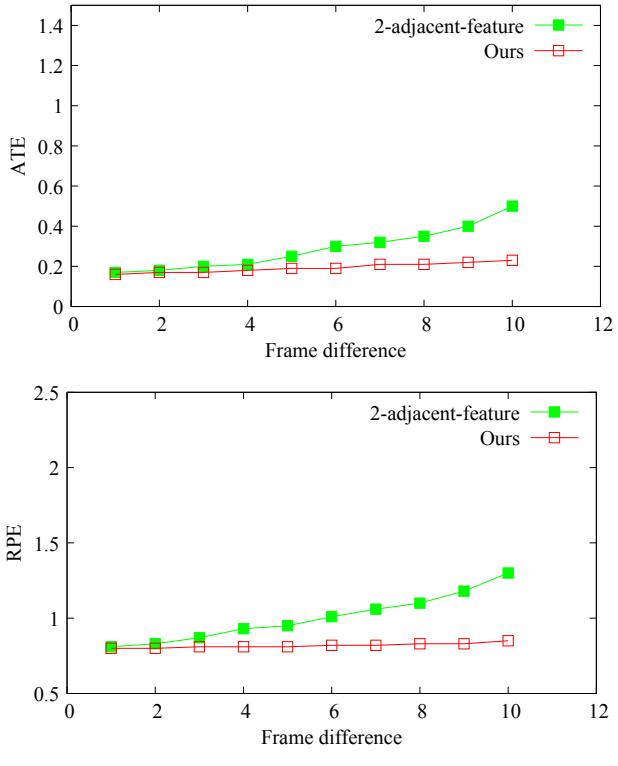


Fig. 8 Comparison between our method and the consecutive-feature method on fr1/floor for varying frame difference.

5 Conclusions and future work

This paper has proposed a novel feature-based camera pose optimization algorithm which efficiently and robustly estimates camera pose in online RGB-D reconstruction systems. Our approach utilizes the feature correspondences from all previous frames and optimizes camera poses across frames. We have implemented our method within two state-of-the-art online RGB-D reconstruction platforms. Experimental results verify that our method improves current online systems in estimating more accurate camera poses and generating more reliable reconstructions for RGB-D data with large shifts between consecutive frames.

Considering that our camera pose optimization method is only part of the RGB-D reconstruction system pipeline, we aim to develop a new RGB-D reconstruction system with our camera pose optimization framework in it. Moreover, we will also explore utilizing our optimized camera poses in previous frames to update the previously reconstructed model in the online system.

References

[1] Keller, M.; Lefloch, D.; Lambers, M.; Izadi, S.; Weyrich, T.; Kolb, A. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In: Proceedings of the International Conference on 3D Vision, 1–8, 2013.

[2] Nießner, M.; Zollhöfer, M.; Izadi, S.; Stamminger, M. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics* Vol. 32, No. 6, Article No. 169, 2013.

[3] Zhou, Q.-Y.; Koltun, V. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 155, 2014.

[4] Newcombe, R. A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A. J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality, 127–136, 2011.

[5] Whelan, T.; Kaess, M.; Johannsson, H.; Fallon, M.; Leonard, J.; McDonald, J. Real-time large-scale dense RGB-D slam with volumetric fusion. *The International Journal of Robotics Research* Vol. 34, Nos. 4–5, 598–626, 2015.

[6] Peasley, B.; Birchfield, S. Replacing projective data association with Lucas–Kanade for KinectFusion. In: Proceedings of the IEEE International Conference on Robotics and Automation, 638–645, 2013.

[7] Henry, P.; Krainin, M.; Herbst, E.; Ren, X.; Fox, D. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research* Vol. 31, No. 5, 647–663, 2012.

[8] Newcombe, R. A.; Lovegrove, S. J.; Davison, A. J. DTAM: Dense tracking and mapping in real-time. In: Proceedings of the IEEE International Conference on Computer Vision, 2320–2327, 2011.

[9] Whelan, T.; Leutenegger, S.; Salas-Moreno, R.; Glocker, B.; Davison, A. ElasticFusion: Dense SLAM without a pose graph. In: Proceedings of Robotics: Science and Systems, 11, 2015.

[10] Whelan, T.; Johannsson, H.; Kaess, M.; Leonard, J. J.; McDonald, J. Robust real-time visual odometry for dense RGB-D mapping. In: Proceedings of the IEEE International Conference on Robotics and Automation, 5724–5731, 2013.

[11] Zhang, K.; Zheng, S.; Yu, W.; Li, X. A depth-incorporated 2D descriptor for robust and efficient 3D environment reconstruction. In: Proceedings of the 10th International Conference on Computer Science & Education, 691–696, 2015.

[12] Endres, F.; Hess, J.; Engelhard, N.; Sturm, J.; Cremers, D.; Burgard, W. An evaluation of the RGB-D SLAM system. In: Proceedings of the IEEE International Conference on Robotics and Automation, 1691–1696, 2012.

[13] Huang, A. S.; Bachrach, A.; Henry, P.; Krainin, M.; Maturana, D.; Fox, D.; Roy, N. Visual odometry and mapping for autonomous flight using an RGB-D camera. In: *Robotics Research*. Christensen, H. I.; Khatib, O.; Eds. Springer International Publishing, 235–252, 2011.

[14] Xiao, J.; Owens, A.; Torralba, A. SUN3D: A database of big spaces reconstructed using SfM and object labels. In: Proceedings of the IEEE International Conference on Computer Vision, 1625–1632, 2013.

[15] Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. G^2o: A general framework for graph optimization. In: Proceedings of the IEEE International Conference on Robotics and Automation, 3607–3613, 2011.

[16] Roth, H.; Vona, M. Moving volume KinectFusion. In: Proceedings of British Machine Vision Conference, 1–11, 2012.

[17] Zeng, M.; Zhao, F.; Zheng, J.; Liu, X. Octree-based fusion for realtime 3D reconstruction. *Graphical Models* Vol. 75, No. 3, 126–136, 2013.

[18] Whelan, T.; Johannsson, H.; Kaess, M.; Leonard, J. J.; McDonald, J. Robust tracking for real-time dense RGB-D mapping with Kintinuous. Computer Science and Artificial Intelligence Laboratory Technical Report, MIT-CSAIL-TR-2012-031, 2012.

[19] Henry, P.; Fox, D.; Bhowmik, A.; Mongia, R. Patch volumes: Segmentation-based consistent mapping with RGBD cameras. In: Proceedings of the International Conference on 3D Vision, 398–405, 2013.

[20] Chen, J.; Bautembach, D.; Izadi, S. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics* Vol. 32, No. 4, Article No. 113, 2013.

[21] Steinbrucker, F.; Kerl, C.; Cremers, D. Large-scale multiresolution surface reconstruction from RGB-D sequences. In: Proceedings of the IEEE International Conference on Computer Vision, 3264–3271, 2013.

[22] Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 573–580, 2012.

[23] Stückler, J.; Behnke, S. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation* Vol. 25, No. 1, 137–147, 2014.

[24] Zhou, Q.-Y.; Koltun, V. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics* Vol. 32, No. 4, Article No. 112, 2013.

[25] Zhou, Q.-Y.; Miller, S.; Koltun, V. Elastic fragments for dense scene reconstruction. In: Proceedings of the IEEE International Conference on Computer Vision, 473–480, 2013

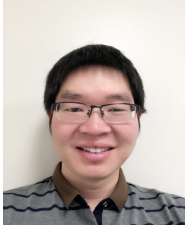
[26] Choi, S.; Zhou, Q.-Y.; Koltun, V. Robust reconstruction of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5556–5565, 2015.

[27] Steinbrücker, F.; Sturm, J.; Cremers, D. Real-time visual odometry from dense RGB-D images. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, 719–722, 2011.

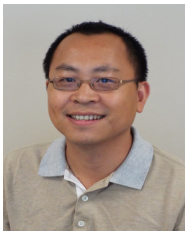
[28] Hänsch, R.; Weber, T.; Hellwich, O. Comparison of 3D interest point detectors and descriptors for point cloud fusion. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* Vol. 2, No. 3, 57, 2014.

[29] Juan, L.; Gwun, O. A comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing* Vol. 3, No. 4, 143–152, 2009.

[30] Yan, W.; Shi, X.; Yan, X.; Wan, L. Computing openSURF on openCL and general purpose GPU. *International Journal of Advanced Robotic Systems* Vol. 10, No. 10, 375, 2013.
[31] Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.
[32] Arun, K. S.; Huang, T. S.; Blostein, S. D. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. PAMI-9, No. 5, 698–700, 1987.



Chao Wang is currently a Ph.D. candidate in the Department of Computer Science at the University of Texas at Dallas. Before that, he received his M.S. degree in computer science in 2012, and B.S. degree in automation in 2009, both from Tsinghua University. His research interests include geometric modeling, spectral geometric analysis, and 3D reconstruction of indoor environments.



Xiaohu Guo received his Ph.D. degree in computer science from Stony Brook University in 2006. He is currently an associate professor of computer science at the University of Texas at Dallas. His research interests include computer graphics and animation, with an emphasis on geometric modeling and processing, mesh generation, centroidal Voronoi tessellation, spectral geometric analysis, deformable models, 3D and 4D medical image analysis, etc. He received a prestigious National Science Foundation CAREER Award in 2012.