Computer-Aided Design 78 (2016) 107–117

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

A multi-frame graph matching algorithm for low-bandwidth RGB-D SLAM☆

Shuai Zheng [a], Jun Hong [a], Kang Zhang [b], Baotong Li [a], Xin Li [b,*]

[a] State Key Laboratory for Manufacturing Systems Engineering, School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, 710049, PR China

[b] School of Electrical Engineering & Computer Science (EECS), Louisiana State University, Baton Rouge, LA 70803, USA

ARTICLE INFO

Keywords:
Multi-frame graph matching
Partial matching
Low-bandwidth SLAM
RGB-D reconstruction

ABSTRACT

This paper presents a novel multi-frame graph matching algorithm for reliable partial alignments among point clouds. We use this algorithm to stitch frames for 3D environment reconstruction. The idea is to utilize both descriptor similarity and mutual spatial coherency of features existed in multiple frames to match these frames. The proposed multi-frame matching algorithm can extract coarse correspondence among multiple point clouds more reliably than pairwise matching algorithms, especially when the data are noisy and the overlap is relatively small. When there are insufficient consistent features that appeared in all these frames, our algorithm reduces the number of frames to match to deal with it adaptively. Hence, it is particularly suitable for cost-efficient robotic Simultaneous Localization and Mapping (SLAM). We design a prototype system integrating our matching and reconstruction algorithm on a remotely controlled navigation iRobot, equipped with a Kinect and a Raspberry Pi. Our reconstruction experiments demonstrate the effectiveness of our algorithm and design.

1. Introduction

Given a set of point clouds sequentially scanned from a 3D scene, to match and stitch these partially overlapped point data and reconstruct the entire scene is a fundamental problem in computer graphics, reverse engineering, and robotic vision. A direct application is the famous Simultaneous Localization And Mapping (SLAM) problem where a robot equipped with a range scanning sensor can navigate around an unknown environment to reconstruct the surrounding and locate its own position. Professional *outdoor* SLAM systems often use expensive LIDAR laser cameras mounted on a vehicle for the urban scanning and mapping. For *indoor* SLAM, in contrast, smaller and cheaper sensors such as Kinect [1] and PrimeSense [2] can be used instead. These outdoor/indoor range scanning cameras often capture not only color images, but also depth information of the scene. The produced RGB-D image sequences, combining pixel-wise color and depth information, allow us to more easily match correlated frames, transform and stitch all the scans into a global coordinate system, and reconstruct the surrounding 3D environment.

In this project, we design an indoor prototype SLAM system, using a mobile iRobot to navigate and map an unknown environment. The iRobot can either move randomly or be remotely controlled. We mount on this iRobot a Kinect sensor, which continuously acquires RGB-D image data of the surrounding environment. In general, several possible approaches can be adopted to process these data for SLAM and environment reconstruction. (1) One is to let the robot carry a powerful-enough processing unit, e.g., a laptop, during its navigation. Then the acquired camera data can be directly processed locally [3]. However, by this approach, the size and cost of the robot will increase significantly, making the system not suitable for narrow corridors or hazardous environments (e.g. with flooding floors); also, the extra weight of the (laptop) processor often takes up most of the load capacity of the robot and makes it energy-inefficient. (2) The second approach is to just let the robot carry a hard disk to save the scan data. The data will be processed after the robot returns and the data in the disk are extracted. However, with this approach, we are not able to simultaneously monitor the robot and control its movement. In addition, to navigate inside a complex and unknown environment without remote human control, the system needs an effective real-time autonomous path planning scheme, which is highly challenging and again requires a powerful processor to be carried by the robot. (3) A third approach is to use an integrated system to obtain data from the camera and transfer them to the control center through a wireless network. The integrated system can be a small and inexpensive

☆ This paper has been recommended for acceptance by Scott Schaefer and Charlie C.L. Wang.

* Corresponding author.
E-mail address: xinli@lsu.edu (X. Li).

http://dx.doi.org/10.1016/j.cad.2016.05.009
0010-4485/

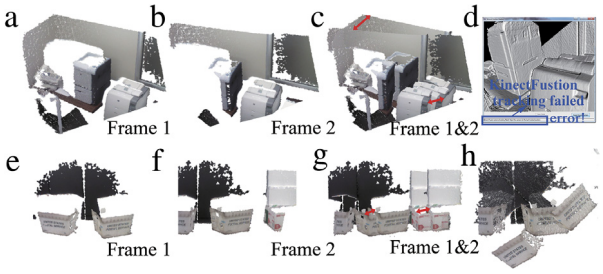

Fig. 1. Kinect-fusion (a–d) and ICP-based (e–h) algorithms fail to stitch frames when camera shift is big (i.e., the overlap between frames is small). (a–b) are the two frames to match, (c) is a global view of the offset between the two frames in a same coordinate, and (d) shows a snapshot where the Kinect-fusion program [5] fails in matching these frames. (e–g) illustrate another similar scene with offset of two frames, and (h) shows undesirable stitching result based on pairwise matching using SIFT+RANSAC [9].

chip such as Raspberry Pi [4], upon which data transfer and robot control can be handled easily. With this design, the data can be processed remotely, and a user can (nearly-) simultaneously monitor and control the robot to finish the navigation and mapping. In our experiments, we adopt this third approach in our design of robotic navigation and mapping. Whichever way the robot is designed, a *key geometric modeling* problem to tackle is how to reliably match sequential geometric data sets obtained by the robot. Developing such a reliable matching algorithm is the focus of this paper.

Considering the sequentially acquired RGB-D data sets, Kinect-fusion [5] and its variants [6–10] are popular algorithms used for real-time 3D reconstruction. In these algorithms, the acquired RGB-D images are sequentially aligned with the previous frame using variants of the iterative closest point (ICP) algorithm; GPUs are fully utilized to accelerate the processing speed for real-time performance.

However, a key assumption of these algorithms is that *the acquisition frame rate is high and adjacent frames do not shift a lot*. If a big camera shift exists between consecutive shots, these algorithms are prone to fail, because ICP-based matching easily gets trapped by local minima. Fig. 1(a–d) shows such an example. Furthermore, efficiently handling such data may not be easily achieved by a practical cost-efficient mobile-working system: the wireless connection is often not fast enough to support communication in such a high frame-rate speed, and matching data in a 30fps rate requires a powerful processor with advanced GPUs.

Another type of approach [11–14] is to extract a set of features, then match them, rather than matching all the points, from different frames. The general pipeline of such feature-matching based methods has four steps: feature detection, feature descriptor computation, feature mapping, and transformation estimation. Pairwise matches can also be globally refined to reduce accumulated errors or ensure certain groupwise geometric consistency [15–17]. These approaches model and prune many pairwise matchings together, and hence, can work more reliably under noisy or small-overlap scenarios. However, they do not handle sequential streaming data well due to the high computational complexity. Although many geometric matching algorithms have been developed in recent years [18], with the decrease on data acquisition frame rate, the overlap between two frames becomes smaller and smaller. This still poses significant challenge to the current partial matching algorithms. Fig. 1(e–h) illustrate a failure example when matching two frames with relatively small overlap using a SIFT–RANSAC matching algorithm [9].

Based on above observations, we believe developing a novel partial matching algorithm, which could more reliably align *noisy* data frames undergoing significant camera shift (hence, correlated frames only have small overlap regions), will greatly benefit the practical reconstruction from dynamic robotic environment scan. In this paper, we propose a new algorithm for more robust matching of noisy and small-overlapped point cloud data sets. The *main contributions* of this paper include:

1. a novel multi-frame graph matching algorithm to map noisy data sets with relatively small overlaps;
2. an inexpensive robotic prototype system using iRobot, Raspberry Pi, and Kinect sensors, which demonstrates our matching algorithm's application on 3D indoor environment mapping.

2. Related work

We briefly review closely related work on 3D reconstruction from sequential RGB-D data, and refer the readers to the survey papers [19,20]. There are two general reconstruction approaches: (1) *Dense matching* methods, which analyze all points/pixels between two frames based on their geometric and/or photometric characters.

(2) *Feature matching* methods, which first solve a coarse correspondence among features in different frames then compute inter-frame alignment based on this coarse correspondence.

2.1. Dense matching approaches

One branch of environment reconstruction is to utilize all pixels in the current RGB-D frame to match with the previous frame, also known as Dense-SLAM. Kerl et al. [21] computed photometric characters from RGB frame and geometric characters from Depth frame between every two frames to get camera positions. However, the high requirements of photometric consistency limit the baseline of the matches, typically narrower than those that features matching algorithms allow. This has a great impact in reconstruction accuracy, which requires wide baseline observations to reduce depth uncertainty. Also, they are quite affected by rolling-shutter, auto gain, and auto exposure artifacts. To enhance the performance, [22,23] perform offline analysis of camera trajectories to achieve dense scene reconstruction and high fidelity texture mapping. However, these approaches need to run off-line for hours using parallel hardware, and thus are not suitable for a low-cost robot carried sensing and computing device.

In the software Kinect fusion [5] developed by Microsoft, consequent frames are stitched using a GPU accelerated ICP algorithm. The nearest correspondences of all the points in the RGB-D data are iteratively calculated and used to refine the transformation between frames. Whelan et al. [7] implemented an RGBD based ICP and implemented it in GPU, which is an enhancement of the original depth data based ICP. Nießner et al. [10] employ an inertial measurement unit (a gyroscope embedded in Kinect) to record camera pose, and to decide ICP initial position and reduce the number of ICP iterations needed in stitching RGB-D frames. Another effective invariant of Kinect fusion based dense matching method for indoor scene reconstruction, suggested by Zhang et al. [24], employs surface fitting on point clouds to detect flat planar patches which are the major salient structures exhibited in an indoor environment. This algorithm performs desirably in reconstructing noisy Kinect scans for large indoor scenes. These approaches often require a powerful graphic hardware to carry the parallel computation or need an assisting hardware to adjust the error generated by the ICP algorithm [25,26]. This usually significantly increases the cost of the robot. Furthermore, alignment based on ICP converges to local optimum near the initial alignment, hence, it is not robust when handling large inter-frame motion or planar surface data, and will result in incorrect stitching or visual artifacts in practical reconstruction from scans with low frame rates [27].

2.2. Feature matching approaches

Features are points having salient geometric or texture properties. Instead of dense matching, feature matching approaches first extract a set of features, then solve correspondences of features and calculate the transformation based on that. Two components dictate the effectiveness of a feature matching algorithm: *feature detection and description*, and *feature mapping*.

Feature detection and description. In RGB-D data processing literature, popular 3D features can be classified into the *RGB-based* and *geometry-based*. *RGB based detectors* are widely used in image registration [28]. When a 3D point cloud is equipped with RGB (or grayscale) information, then features can firstly be detected in the RGB or grayscale channel, then map to their corresponding points in 3D [29,9]. For RGB-D data acquired by Kinect, however, *motion blur* and *rolling shutter* occur frequently. Therefore, directly using color information and RGB-based features to match Kinect scan data is sometimes unreliable. *Geometry-basedbedkowski2012real detectors* describe shapes using their depth information and local geometry. The Gaussian pyramid is usually used in creating a multi-scale space, facilitating the generalization of 2D descriptors for RGB-D data. Such detectors include *2.5D SIFT* [26,30], *3D SIFT* [31], *3D SURF* [32], etc. However, building a Gaussian pyramid is often expensive and these detectors often result in thousands of keypoints, making the subsequent computation prohibitively expensive. Without building a multi-scale space, one can also use local keypoints in partial matching computation. Effective detectors such as *3D Harris* [33], *Locality Sensitive Hashing* [34], and *Intrinsic Shape Signature* [35] are recently developed, based on local geometric characteristics: they fit each point with its neighbors into a quadratic surface, and decide whether it is a keypoint by the surface's mathematical description. Since the Kinect scan data are very noisy, local descriptors could become unreliable, hence, a preprocessing surface smoothing is often needed. However, after smoothing, these detectors often result in very few extracted keypoints when the surfaces are flat or lack salient geometric variance in the scene.

The *Normal Aligned Radial Feature (NARF)* [36] computes range images from a point cloud, then extracts characteristic object borders and corner points as keypoints. NARF has several advantages in modeling features in scan data from Kinect. (1) It utilizes depth range image and its geometric information, hence, is robust against the motion blur issue in RGB based detectors. (2) It avoids the building of multi-scale space, and so is computationally efficient. (3) It is viewpoint-invariant, and less sensitive to outliers or local geometric perturbation; keypoints extracted from the borders and corners of the range image can better describe the silhouette structure of the objects.

Feature mapping is the procedure to establish a bijective correspondence between features on different frames. In the reconstruction problem we consider here, the Kinect scans are noisy, possessing many outliers, and the overlap between frames is small due to the limited acquisition frame-rate. Hence, the outliers often outnumber the inliers, and a robust feature mapping algorithm is needed in such scenarios. Feature mapping algorithms developed to tackle outliers generally include *voting* [37], *RANSAC* [38], *forward search* [39], and *graph matching* [40].

The *Voting-based* methods like [37] allow each potential feature pair to "vote" for its transformation model and the transformation with the majority votes is the result. This voting strategy can handle small number of outliers, but its success relies on the assumption that the inliers are the majority. When the outliers become the majority, the *RANSAC* strategy [38] could have better robustness. RANSAC estimates solutions on randomly sampled subsets. When a subset results in a solution whose score passes a threshold, this solution is accepted. The RANSAC algorithm is more robust than voting against outliers. But when outliers dominate the scene, many random subsets need to be generated before a correct solution can be found, so the algorithm becomes slow and unreliable. Henry et al. [12] introduce a RGB-D ICP algorithm based on RANSAC. It firstly detects SIFT keypoints, then performs a RANSAC to filter the keypoint correspondences and get the initial rigid transformation. Then an ICP algorithm is performed to refine the transformation. Endres et al. [19] propose a RANSAC based matching algorithm. Using SIFT or SURF as keypoint detectors, they further incorporate loop closure constraints and solve it by a graph optimization, which greatly reduces the accumulating error. These RANSAC-based algorithms require the offsets between frames to be small: with large displacement frames, the ratio of outliers over inliers could become very big and the result will be unreliable. The *Forward search* method [39] is designed to handle large-sized problems in the presence of outliers. It builds an initial set of correspondences, then iteratively refines the matching based on the inliers inside this set and simultaneously updates inliers based on the refined matching. In many cases when the outliers are not dominant, the forward search can be more efficient and more accurate than the RANSAC, due to its iterative refinement strategy. Unfortunately, if outliers are dominant in all the extracted features, starting a forward search with reliable initial inlier sets is not guaranteed. Also, the large number of outliers could easily affect this deterministic search and yield an incorrect solution produced by the majority of the features.

The *Graph matching* methods [40–42] construct a graph whose vertices denote corresponding feature pairs and edges encode their spatial relations. Based on the assumption that correct corresponding feature pairs are mutually coherent while incorrect pairs are not, a graph matching algorithm extracts the maximal coherent cluster from a pre-computed affinity matrix, and converts the feature correspondence problem to a quadratic integer programming problem, utilizing both feature similarity and geometric spatial consistency among features' locations. The graph matching can often handle large number of outliers more reliably than RANSAC and forward search. High-order graph matching [43,44] has been studied to improve the reliability of pairwise matching. These algorithms use higher order primitives such as triangles or quadrilaterals (formed by multiple features), instead of lines (formed by two features) to reduce ambiguous mismatching caused by outliers and improve the matching reliability. To better circumvent local minima in integer quadratic programming when solving the graph matching, Cho et al. [42] develop a reweighted random walk stochastic optimization scheme to find the approximate solution that is more resistant to deformation and feature outliers. Leng et al. [45] add a perturbation to the eigenvector of the Laplacian matrix and utilize the characteristic that small eigenvalues are sensitive to perturbations, while large eigenvalues are relatively stable, also producing more reliable global solutions to graph matching with respect to noisy scenes and structural corruption. However, when many outliers exist, there is a certain chance that false correspondences are spatially coherent with each other, yielding higher spatial consistency score than true correspondences (see Section 3.6). In such a case pairwise graph matching may also become unreliable. How to more reliably extract correspondence from noisy and lightly overlapped data sets still remains a challenging problem.

3. Multi frame graph matching

3.1. Basic idea and algorithm overview

2-frame matching model. The *graph matching* algorithm [40] and its variants [41,42] compute a bijective node-to-node assignment

between two graphs G and G' by solving a constrained quadratic integer optimization problem: (1) identify possible corresponding node pairs $\mathbf{c}_k = (v_i, v'_j)$, $v_i \in G$, $v'_j \in G'$ (where v_i, v'_j represent feature points in graph G and G'); (2) construct an affinity matrix M, whose diagonal elements m_{kk} encode similarity between descriptors of v_i and v'_j in node pair $\mathbf{c}_k$, and non-diagonal elements m_{rs} measure spatial geometric coherency between node pairs c_r and c_s; (3) maximize $f(\mathbf{x}) = \mathbf{x}^T M \mathbf{x}$ subject to bijectively constraints, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the n-dimensional indicator vector to solve: $x_i = 1$ if the pair c_i is in the final assignment and $x_i = 0$ otherwise.

N-frame matching model. We can generalize the above 2-frame model to maximize spatial consistency of corresponding features in the matching of N frames. Given the N frames $F_1, F_2, \ldots, F_n$, we construct a graph $G = (V, E)$, where each node $\mathbf{c}_i = (v_{1i_1}, v_{2i_2}, \ldots, v_{Ni_N})$ is an N-correspondence tuple, $v_{1i_1} \in F_1$, $v_{2i_2} \in F_2$, $v_{Ni_N} \in F_N$. We denote it as $\mathbf{c}_i = (i_1, i_2, \ldots, i_N)$ for simplicity. An edge $e_{ij} = (\mathbf{c}_i, \mathbf{c}_j) \in E$ is constructed if the correspondence tuples $\mathbf{c}_i$ and $\mathbf{c}_j$ are spatially consistent to each other. A weight function $w(e_{ij})$ defined on e_{ij} measures the spatial consistency between $\mathbf{c}_i$ and $\mathbf{c}_j$. Intuitively, if frames differ by rigid transformations, then the *exact spatial consistency* means the distance between corresponding points should be preserved, namely, $|v_{1i_1} - v_{1j_1}| = |v_{2i_2} - v_{2j_2}| = \cdots = |v_{Ni_N} - v_{Nj_N}|$. Details on the design of consistency metric are discussed in Section 3.5.

N-frame graph matching algorithm. With the generalized matching model, our idea is that instead of just iteratively stitching a new frame to its previous frame, we try to correlate N frames at the same time, where $N > 2$ is a predetermined number. This will make the algorithm more reliable against noise and ambiguous alignment due to small overlap among frames. On the other hand, requiring all the corresponded features always appear in N consecutive frames is sometimes not easy. The overlapped regions of scenes and available salient features vary during the camera movement. In our implementation, we require at least $\eta_L = 10$ consistent feature points to be obtained in all the N frames. Thus, first, the frame number N should not be too big. Second, our following algorithm adaptively degenerates into the matching using smaller number of frames. Our algorithm can be summarized as follows.

In: Parameters N and η_L;
(1) When a new frame F_i is obtained: if $i > N$ then $k = N$, else $k = i$;
(2) Extract keypoints on F_i and compute their descriptors (Section 3.2);
(3) IF $k > 1$, try to match F_i with its previous $(k-1)$ frames:
  (3.1) Extract initial corresponding k-tuples, $\mathcal{L}_k^0$, using KNN (Section 3.3);
  (3.2) IF $|\mathcal{L}_k^0| < \eta_L$, THEN no enough tuples found, $k = k - 1$, GOTO (3);
  (3.3) ELSE solve k-frame graph matching to get the final correspondence $\mathcal{L}_k$ (Section 3.4);
  (3.4) Calculate the rigid transformation by $\mathcal{L}_k$ and transform F_i accordingly.
(4) IF this is not the last frame, THEN $i = i + 1$ and GOTO (1), ELSE STOP.

In this algorithm, N indicates the number of frames we want the multi frame graph matching to start with. As discussed above, the size should be moderate. In our experiments we found $N = 3$ or 4 is a practically suitable number, which maintains a good balance between available feature numbers and matching reliability. η_L determines during k-frame graph matching, how many initial corresponding k-tuples are considered enough to run a robust graph matching. We simply set $\eta_L = 10$; and in all our experiments, $|\mathcal{L}_k^0| > 10$ satisfies easily for $k = 2$. In the worst case, even between two consecutive frames there are no enough features for pairwise matching, then simply stop matching frames and restart over, setting the current frame as the new first frame. This is like when $i = 1$ and no matching for F_1 is computed.

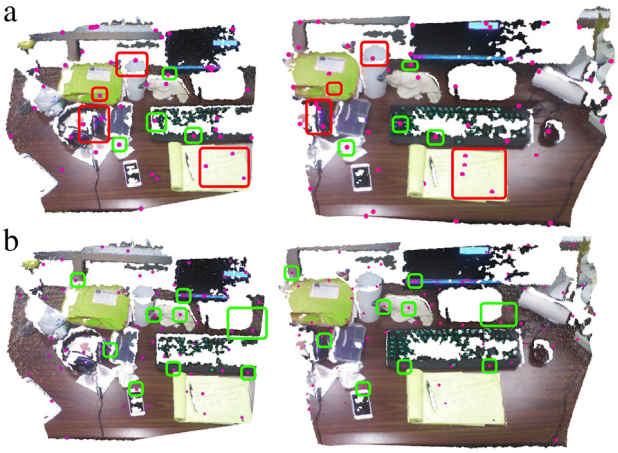

Fig. 2. Comparisons between SIFT3D and NARF, with correct repeatable keypoints in green and wrong ones in red: (a) Some features extracted by SIFT3D detector are not repeated in both frames. (b) NARF detector results in better repeatability in feature detection. To help visualize the correspondence, we add a few rectangles to highlight the corresponding features. Green and red rectangles indicate correct and incorrect correspondences, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.).

3.2. Feature extraction

Using features reduces the amount of points for which similarity computation has to be performed, hence, it significantly reduces the number of correspondences that we need to examine. The challenging issue in feature detection from Kinect scans is keypoint robustness against noise. Here we choose the Normal Aligned Radial Feature (NARF) keypoint detector [36], which was reported to be efficient and reliable in describing features in noisy point cloud. Extracting NARF keypoints consists of six steps: (1) take the input 3D point cloud and convert it to a *spherical* range image (i.e., three 2D range images), (2) perform a heuristic-based detection of object borders, (3) compute normals for border points, (4) compute principal curvature for non-border points, (5) compute interest value for all points, (6) isolate keypoints. Please refer to Steder et al. [36] for details.

We observed several desirable properties of the NARF detector for our problem here: (1) NARF creates a range image of the point cloud. It does not require a Gaussian pyramid creation, and hence is very efficient. (2) NARF takes object borders into account, which arise from view dependent noncontinuous transitions from the foreground to the background. Thus, the silhouette of an object can be detected robustly from the range image, even with big view change. (3) Directly built upon the characteristics of borders' geometry and normal, the descriptor of NARF is natural and more effective than many artificially combined keypoint detectors and feature descriptors in describing noisy scenes. A recent comparative study [46] shows that NARF and 3D Sift keypoints are representative fixed-scale and multi-scale keypoint detectors dealing with indoor and noisy Kinect scan. Keypoint *quantity* and *detection time* are documented in the paper, indicating 3D SIFT is significantly slower and results in many more keypoints than NARF, which will introduce significant extra computation complexity of the subsequent matching computation. We also perform a comparison on *repeatability* of these two keypoint detectors. As shown in Fig. 2, the 3D SIFT detector misses quite a few salient matchable features in the second scene; in contrast, NARF results in keypoints with better repeatability. Through these observations and experiments, due to the better behaviors of NARF [36] in partial matching of noisy data, in this work, we utilize it as our keypoint detector.

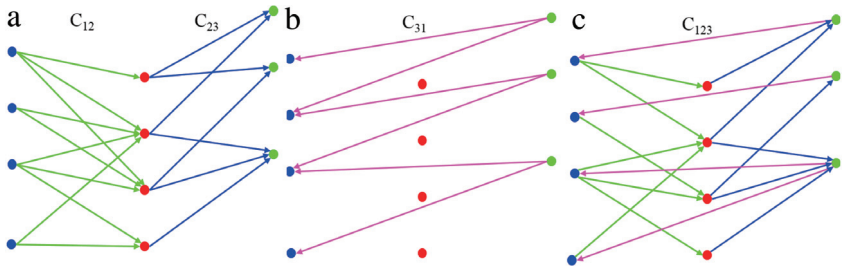

Fig. 3. Using directed loops to extract corresponding 3-tuples from KNN. (a) KNN correspondence set $\mathbf{C_{12}}$ between features from frame 1 and frame 2, and respectively, correspondence set $\mathbf{C_{23}}$ between features from frames 2 to 3. (b) The correspondence set $\mathbf{C_{31}}$ defined by KNN of features from frames 3 to 1. (c) Each *(green, blue, red)* triangle indicates a corresponding triplet $\mathbf{C_{123}}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.).

3.3. Initial correspondence

Given N frames, suppose from frame j we extract n_j feature points, denoted as $P_j = \{p_{j,i_j}, i_j = 1, \ldots, n_j\}$. Their NARF descriptors are histograms H_{ji_j}, respectively. We establish an initial correspondence among keypoints from different frames using a K-Nearest Neighbor (KNN) matcher using the L_2-norm between H_{j,i_j} and $H_{j+1,i_{j+1}}$.

Given two keypoints p_i and p_j, we compute their NARF descriptors H_i and H_j which are 36-dimensional vectors with each element ranging from -1 to 1, then measure their L_2-norm similarity:

$$d(H_i, H_j) = 1 - \|H_i - H_j\| = 1 - \frac{\sqrt{\sum_{s=1}^{36}(H_i^s - H_{i'}^s)^2}}{36}. \quad (1)$$

We observe from our experiments that a typical Kinect scanned indoor frame usually contains about 200 NARF keypoints. So we set $K = 10$ for the KNN matcher in all our reconstructions. Therefore, each keypoint's 10 most similar keypoints are kept.

If there are N correspondence pairs extracted by KNN, that $\mathbf{c}_{1,2} = (p_{1i_1}, p_{2i_2}), \ldots, \mathbf{c}_{n_1,n} = (p_{(N-1)i_{N-1}}, p_{Ni_N})$, and $\mathbf{c}_{n,1} = (p_{Ni_N}, p_{1i_1})$, we say they form a *directed loop*. The elements in one loop are likely to correspond to one feature point, and we create an N-tuple $\mathbf{c}_{i_1 \ldots i_N} = (p_{1i_1}, p_{2i_2}, \ldots, p_{Ni_N})$, or called a *consistent N-tuple*, to encode such a potentially consistent feature.

Using KNN, we first compute all the corresponding feature pairs between frames j and $h = j + 1, L_{j,h} = \{(p_{ji_j}, p_{hi_h})\}$. Then, extract all the index N-tuples $(i_1, i_2, \ldots, i_N)$ such that $(p_{1i_1}, p_{2i_2}) \in L_{1,2}$, $(p_{2i_2}, p_{3i_3}) \in L_{2,3}, \ldots, (p_{(N-1)i_{N-1}}, p_{Ni_N}) \in L_{N-1,N}$, and $(p_{Ni_N}, p_{1i_1}) \in L_{N,1}$. A set of corresponding N-tuples is extracted, and it forms an initial correspondence set $\mathcal{L}^0$ (which will be refined in the subsequent N-frame graph matching). Fig. 3 illustrates an example of this process where $N = 3$. In (c), each triangle (formed by red, green, and blue edges) indicates a corresponding 3-tuple in the initial correspondence set.

3.4. Multi-frame graph matching

From the initial set of corresponding N-tuples $\mathcal{L}^0 = \{\mathbf{c}_i = (i_1, i_2, \ldots, i_N)\}$, we solve an optimization problem to extract a subset $\mathcal{L} \subset \mathcal{L}^0$ which maximizes (1) similarity among features in each selected N-tuple, and (2) mutual spatial consistency between pairs of N-tuples in $\mathcal{L}$, subject to a bijectivity constraint. The bijectivity constraint enforces that if a feature p_{ri} in frame r is mapped to another feature p_{sj} in frame s, then it should not be mapped to another feature in frame s. We construct a graph, then an associate affinity matrix, to solve this optimization.

3.4.1. Graph construction

We construct a graph $G = (V, E)$ as follows. A node $\mathbf{c}_i \in V$ is an N-tuple in $\mathcal{L}^0$, and on each edge $e_{ij} \in E$ we use a weight function $w_{ij} = S(\mathbf{c}_i, \mathbf{c}_j)$ to describe the spatial consistency between two correspondence N-tuples $\mathbf{c}_i = (i_1, i_2, \ldots, i_N)$ and $\mathbf{c}_j = (j_1, j_2, \ldots, j_N)$. Here since the transformation between two point clouds is rigid, the S function should evaluate how well the geometric shape formed by (i_1, j_1) is preserved after it is transformed to (i_2, j_2), and to (i_N, j_N), etc.

3.4.2. Affinity matrix construction

We define an affinity matrix M from graph G. Each diagonal element $M_{i,i}$ describes the similarity among the N features in these N-tuples $c_i = (p_{1i_1}, p_{2i_2}, \ldots, p_{Ni_N})$. We set $M_{i,i} = D(p_{1i_1}, p_{2i_2}, \ldots, p_{Ni_N})$ to measure similarity of features,

$$D(p_{1i_1}, p_{2i_2}, \ldots, p_{Ni_N}) = \frac{1}{N}(d(H_{p_{1i_1}}, H_{p_{2i_2}}) + \cdots + d(H_{p_{(N-1)i_{N-1}}}, H_{p_{Ni_N}}) + d(H_{p_{Ni_N}}, H_{p_{1i_1}})). \quad (2)$$

A non-diagonal element, $M_{i,j}$, should measure the spatial consistency between two corresponding N-tuples $\mathbf{c}_i$ and $\mathbf{c}_j$. Transformations between point clouds acquired by Kinect are rigid. Therefore, each pair of corresponding features should preserve its distance in different frames. We say two pairs of corresponding features *spatially consistent* if the distance does not change. We measure this spatial consistency value using a function $S(\mathbf{c}_i, \mathbf{c}_j)$, which will be detailed in Section 3.5.

3.4.3. Bijectivity constraint

The bijectivity constraints can be formulated as follows. N-tuples $(i, j, \ldots, k)$ and $(i, j', \ldots, k')$, $(j \neq j', k \neq k')$ should not be selected together. Namely, if point p_{1i_1} corresponds with p_{2j_2}, then it should not be mapped to $p_{2j'_2}$ again. Similarly, $(i, j, \ldots, k)$ should not co-exist with $(i', j, \ldots, k')$ or $(i', j', \ldots, k)$. Therefore, the bijectivity constraint can be formulated as a linear constraint $A\mathbf{x} \leq \mathbf{b}$. A is a sparse matrix consists of 0 and 1 elements. On each row of A, the non-zero elements give the indices of tuples that are associated with a same keypoint in one frame. Therefore, these tuples have mutual conflicts. No more than one among these tuples should be selected in the final solution, as indicated by the vector $\mathbf{b} = (1, 1, \ldots, 1)^T$.

3.4.4. Solving node-to-node assignment

The assignment problem finally reduces to solving an integer quadratic problem:

$$\mathbf{x}_b^* = argmax(\mathbf{x}^T M \mathbf{x}), \quad \text{s.t.} A\mathbf{x} \leq b, \mathbf{x}_b^* \in \{0, 1\}^n, \quad (3)$$

where $A\mathbf{x} \leq b$ is the aforementioned bijectivity constraint, and the final solution $\mathbf{x}_b^*$ should be rounded to only contain 0 and 1 elements.

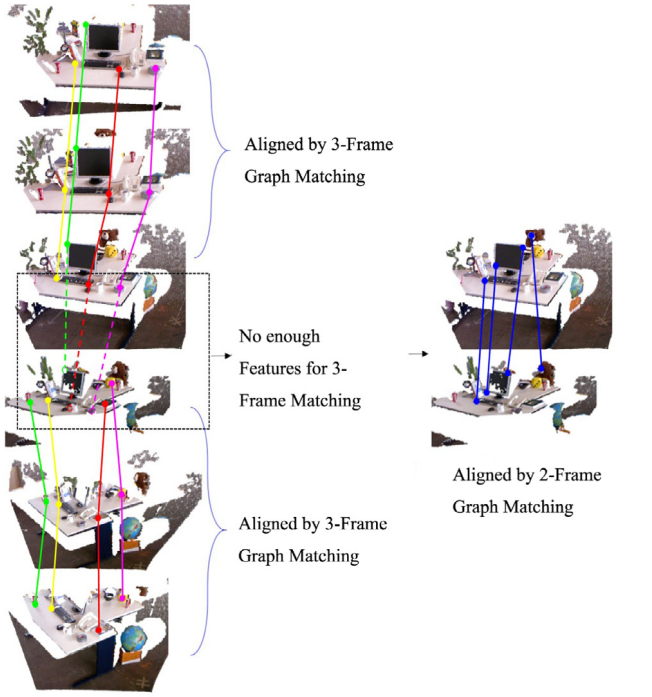

Fig. 4. Applying multi-frame matching (here $N = 3$) to align sequential frames. Some frames are aligned through 3-frame graph matching. But for some frames, coherent features (among them and their previous two frames) are insufficient to support 3-frame matching. Then pairwise (2-frame) graph matching is applied to stitch these frames.

We simply use the spectral matching algorithm from [40] to solve x_b^*: first compute the principal eigenvalue and its corresponding eigenvector, denoted as x^*, then sort elements in x^*, and find the greatest element p and set $x_b^*(p) = 1$. Then, iteratively, following the descending order, find all elements q in x^* that do not have conflict with existing marked elements and set $x^*(q) = 1$, while marking the indicator of each conflict element to 0.

3.4.5. Cross-frame transformation

Finally, when feature correspondence between F_i and its previous frames is obtained, a singular value decomposition can be used to compute the transformation between F_{i-1} and F_i, which is the least square solution of rotation matrix and translation vector [47].

Our multi-frame matching algorithm adaptively decreases the number of frames to match if no enough potential features coherently exist in multiple frames. Fig. 4 illustrates such an example: The 3rd frame in this figure is matched with the first two frames through a 3-frame graph matching, the 4th and 5th frames do not share sufficient features with the previous two frames, so they are stitched through pairwise matchings.

3.5. Measuring spatial consistency

Given two corresponding N-tuples $\mathbf{c}_i$ and $\mathbf{c}_j$, we need to define a function $S(\mathbf{c}_i, \mathbf{c}_j)$ to measure their spatial consistency. In this Kinect SLAM problem from Kinect scans, the transformations between different frames are *rigid*. So $S(\mathbf{c}_i, \mathbf{c}_j)$ should be maximal if $\|p_{1i}p_{1j}\| = \|p_{2i'}p_{2j'}\| = \cdots = \|p_{Ni^N}p_{Nj^N}\|$. And $S(\mathbf{c}_i, \mathbf{c}_j)$ should define a metric to measure the deviation among these distances. Here we propose two metrics, with the second one slightly better.

3.5.1. Metric one: Simple deviations

In the 2-frame graph matching, the spatial consistency score is defined by a deviation of the pairwise distances between point pairs [40].

This idea can now be generalized to match N frames:

$$S_d(\mathbf{c_i}, \mathbf{c_j}) = \begin{cases} C - \dfrac{\sum_{k=1}^{N}(d_{i_kj_k} - d_{i_{k+1}j_{k+1}})^2}{2N\theta_d^2} & \text{if } \forall k, |d_{i_kj_k} - d_{i_{k+1}j_{k+1}}| < 3\theta_d \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $d_{i_kj_k} = \|p_{ki_k}p_{kj_k}\|$ is the distance between the pair of feature points and the index is treated as k modulo N (hence, the last term is $d_{i_Nj_N} - d_{i_1j_1}$). C is a constant simply set to 4.5 in [40], and θ_d is a threshold to filter out edge pairs with significantly different lengths.

3.5.2. A second approach: Scale Jacobian metric

Scale Jacobian is a term commonly used to evaluate the quality of triangular meshes [48]. In a 3-frame matching, we measure aforementioned distance similarity by creating a triangle using the three distances and measure the equality of this triangle. The three distances $d_{ij}, d_{i'j'}, d_{i''j''}$ can be used to construct a 2D triangle: let $(x_1, y_1) = (0, 0)$, $(x_2, y_2) = (d_{ij}, 0)$, and $(x_3, y_3) = (\frac{d_{ij}^2 + d_{i'j'}^2 - d_{i''j''}^2}{2d_{ij}}, \sqrt{d_{i''j''}^2 - \frac{d_{ij}^2 + d_{i'j'}^2 - d_{i''j''}^2}{4d_{ij}^2}})$, where $d_{ij} = \|p_{1i}p_{1j}\|$, $d_{i'j'} = \|p_{2i'}p_{2j'}\|$, $d_{i''j''} = \|p_{3i''}p_{3j''}\|$. Then, the function is defined as follows.

If distances satisfy the triangle inequality $d_{ij} - d_{i'j'} < d_{i''j''} < d_{ij} - d_{i'j'}$, we have Eq. (5) given in Box I.

If $d_{ij}, d_{i'j'}$, and $d_{i''j''}$ do not satisfy the triangle inequality, it is clear the three distances deviate a lot from each other, and we set $S_J(\mathbf{c}_i, \mathbf{c}_j) = 0$.

These 3-frame Scale Jacobian metrics can be generalized to N-frame:

$$S_J(d_1, d_2, \ldots, d_n) = \frac{C * \sqrt[\frac{n+1}{2}]{\frac{n^{(n-1)/2}}{n-2}\prod_{i=1}^{n}\left(\sum_{j=1}^{n} d_j - 2d_i\right)\sum_{j=1}^{n} d_j}}{\sum_{j=1}^{n} d_j^2} \quad (6)$$

where d_k is the Euclidean distance between the keypoint pair in the kth frame. Note here we simplify the subscript from d_{ij} and $d_{i'j'}$ to d_1, d_2 for a more succinct formulation in the above equation.

The generalized spatial consistency can be geometrically depicted as *measuring the deviation of an N-sided polygon from an N-sided equilateral polygon*. Although geometrically, N-sided equilateral polygon does not exist for all the integer numbers N, algebraically, we can always measure the spatial consistency using Eq. (6).

3.5.3. Scale Jacobian metric versus simple deviation metric

Between Scale Jacobian Metric and Simple Deviation Metric, we find the Scale Jacobian in practice can better detect incorrect corresponding pairs. We analyze their responses to the sudden deviation in feature pair distance, which often corresponds to wrong matches. Taking an $N = 3$ (3-frame matching) case as an example, suppose the distances of the three feature pairs are d, d, and $d+\alpha d$, where αd describes a perturbation on one distance. The deviation score S_d and the Scale Jacobian score S_J, respectively, can be expanded as,

$$S_d(d, d, d + \alpha d) = 4.5 - \frac{2 * (\alpha d)^2}{6\theta_d^2}, \quad (7)$$

$$S_J(\mathbf{c}_i, \mathbf{c}_j) = \frac{C * \sqrt{3 * (d_{ij} - d_{i'j'} + d_{i''j''})(d_{ij} + d_{i'j'} - d_{i''j''})(-d_{ij} + d_{i'j'} + d_{i''j''})(d_{ij} + d_{i'j'} + d_{i''j''})}}{d_{ij}^2 + d_{i'j'}^2 + d_{i''j''}^2}. \tag{5}$$

Box I.

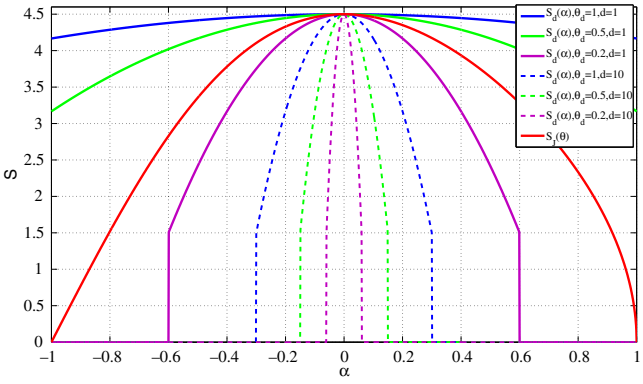

Fig. 5. The responses of S_d and S_J functions on a perturbation αd from the distance d. The red solid curve indicates the Scale Jacobian function, and the blue, green, and purple curves indicate the simple deviation measures when $\alpha = 1.0, 0.5, 0.2$, in which the solid and dotted curves are $S_d(\alpha)$ when $d = 1$ and $d = 10$ respectively. When using a big θ_d (e.g. 1, as suggested by Leordeanu and Herbert [40]), the curve of S_d is flat near the origin and less distinctive than S_J. When using a small θ_d, S_d becomes sharper but also narrower, and hence, more sensitive to noisy feature pairs with big distance.

and

$$\begin{aligned} &S_J(d, d, d + \alpha d) \\ &= \frac{4.5 * \sqrt{3 * (d - \alpha d)(d + \alpha d)(d + \alpha d)(3d + \alpha d)}}{d^2 + d^2 + (d + \alpha d)^2} \\ &= \frac{4.5 * \sqrt{3 * (1 - \alpha^2)(1 + \alpha)(3 + \alpha)}}{2 + (1 + \alpha)^2}. \end{aligned} \tag{8}$$

We illustrate the response functions of S_d and S_J with respect to a perturbation αd in Fig. 5. When using a relatively big θ_d such as 1 suggested by Leordeanu and Herbert [40] or 0.5, the curve of S_d is flat near 0 and less distinctive than S_J. When using a small θ_d, the response function can be made sharper but it also becomes very narrow. For feature pairs that are far away from each other (e.g. their d are big), such a sharp response could be sensitive to noise. Furthermore, this variation in sensitivity shows a more important limitation of S_d: besides the need of tuning θ_d, the sensitivity of response varies for different d values. In a practical scene, against different feature pairs with different lengths, this makes the measure inconsistent. It is difficult to find an always suitable constant θ_d to consistently evaluate feature pairs with different spatial distances. In contrast, the scale Jacobian measure is independent of the scale of d and more consistent. Therefore, in our experiments, we use S_J as the spatial consistency metric.

3.5.4. Tolerance threshold and computational efficiency

In practice, various noise affect the spatial consistency: (1) *scanning error*, (2) *point cloud sampling difference*, and (3) *keypoint repeatability error*. Scanning error is generated by the optics noise of the Kinect camera. Point cloud sampling difference is due to the different sampling locations in the scene that are covered by the camera's pixels. Keypoint repeatability error is produced due to the inconsistency of keypoint detector applied on noisy data noise or due to the ambiguity of shape descriptor. Because of these inevitable errors, the spatial consistency score S_J of even correct correspondences is often slightly smaller than 1 (after normalization, 4.5. The following discussion is based on the function value before normalization). We use a threshold $\eta < 1$ to filter spatially inconsistent pairs. In the affinity matrix, if the spatial consistency score of $(\mathbf{c}_i, \mathbf{c}_j)$ is smaller than η, we consider them to be inconsistent and set $M_{ij} = 0$. With the control of η, a great number of incorrect correspondences are filtered out. This sparsifies the affinity matrix and eliminates small elements. Consequently, (1) the objective function becomes more smooth, having fewer local minima; and (2) the solving of our graph matching becomes numerically more stable and efficient.

We design an experiment to observe the effect of threshold η. A relatively dense keypoint parameter is used to extract more features and a larger initial correspondence set so that the difference of η selection can be better shown. We set η to 0.01, 0.8, and 0.9 out of the full score 1. The results are documented in Table 1: larger η results in faster computational time, but fewer identified correspondence (many not-perfectly-aligned features are rejected). Since in this 3D reconstruction, frames differ by rigid transformations and we do not need many correspondences to uniquely decide the transformation. Therefore, in all our experiments, we set $\eta = 0.9$.

Efficiency of N-frame matching. (1) The computational complexity of the KNN algorithm is $O(K^{N-1} \cdot M)$, where K is the number of nearest neighbors to consider, N and M are the number of frames and average size of keypoints in one frame. When the number of frames to match increases from N to $N + 1$, the computational complexity of KNN increases K times. Since in our experiments, $K = 10-20$, this step is 10 to 20 times slower. However, compared with the graph matching step, this step is less time-consuming and is not the bottleneck. (2) The computational complexity of the graph matching, however, may not increase (especially when the frame-rate is low). This is because the consistent feature tuples obtained in the KNN-computed directed loops usually decreases with the increase of the number of frames to match. Therefore, the number of potential corresponding tuples to be modeled in the multi-frame graph matching, namely, the dimension of the graph matching problem, may not increase. In our experiments, we found that in about a half of scenarios, this dimension remains about the same; while in the other half of scenarios, the dimension either increases or decreases and is decided by both the number of repeated features and the size of overlap, which varies case by case.

3.6. Improved robustness: N-frame vs 2-frame matching

Compared with traditional 2-frame graph matching, the multi-frame graph matching offers more robust alignment for noisy frames relatively small overlap region. This can be justified from the following *two observations*:

(1) Considering N frames in matching is more robust against noise than only considering adjacent 2 frames.
(2) Matching N frames simultaneously is more robust than mutually doing pairwise 2-frame matching in a loop for N times.

Observation 1. Due to the existence of large scan noise, ambiguity of shape descriptors, and the small overlap region, incorrect correspondence inevitably exists in the pairwise graph matching result. Specifically, considering three consecutive frames, suppose feature

Table 1
Tolerance threshold η is related to matching efficiency and accuracy. The experiment is performed on the first 10 frames of data set "fr1 desk" in [19]. The non-zero elements ratio, total computed corresponding N-tuples, and the total graph matching computation time (in seconds) are reported. Bigger η results in a sparser affinity matrix and faster computation; fewer correct correspondences are detected since some corresponding features are rejected. For rigid alignment, a small number of corresponding features is enough, so $\eta = 0.9$ is used in our reconstruction experiments.

η	Non-zero elements ratio	N-tuples computed	T_{GM} (s) for 10 frames
0.01	40.9%	248	48.473 s
0.8	9.1%	175	22.724 s
0.9	5.0%	124	18.073 s

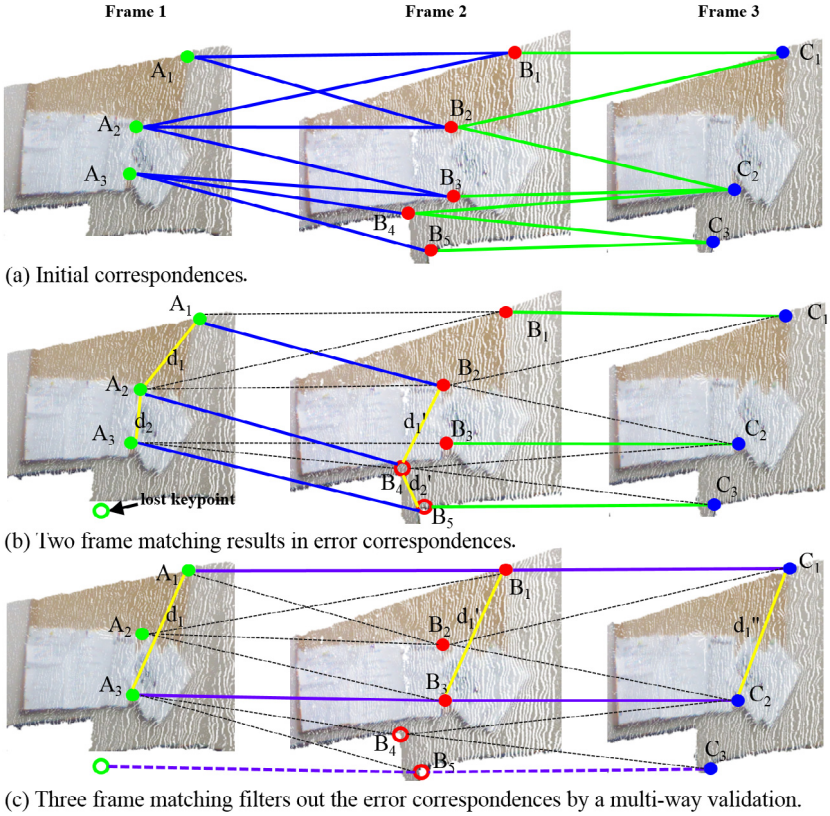

Fig. 6. The initial correspondence (a) is refined after performing a pairwise graph matching, but may still possess incorrect corresponding pairs that are spatially consistent (b). When matching the three frames together, incorrect corresponding pairs are filtered out.

pairs $\mathbf{c}_i = (p_{1i}, p_{2i}, p_{3i})$ and $\mathbf{c}_j = (p_{1j}, p_{2j}, p_{3j})$ are keypoint pairs that are NOT indeed corresponded, then after graph matching, they may be incorrectly matched if (1) p_{ki} and p_{kj} have similar descriptors, (2) their distance $d_k = |p_{ki}p_{kj}|$ between each other remains unchanged in different frames, and (3) their matching (p_{ki}, p_{kj}) can coexist with other correct corresponding pairs without violating the bijectivity constraint. Between two frames k and k', among all selected n corresponding pairs, suppose there are n' pairs that are actually incorrect while satisfying the above (1)–(3) criteria, then $\mu = n'/n \ll 1$. If we verify the above criteria across all the three frames (by either checking them in 3 frames together, or pairwise checking each frame pair for 3 times), then the chance for an incorrect pair to remain in the final selection reduces from μ to μ^3. The algorithm results in a more reliable filtering for the incorrect pairs. Fig. 6 illustrates an example where incorrect corresponding pairs may exist when only pairwise graph matching is performed (b), but they may be filtered out when multiple frames are considered.

Observation 2. To improve the robustness against incorrect corresponding pairs by considering N frames in the matching, we can either (1) compute the pairwise 2-frame matching for N times then filter out inconsistent ones through a directed loop connecting all these frames (which is commonly adopted in existing literature), or (2) compute the matching using all the N frames together (as what we propose here). We observe that the first approach is more likely to result in an undesirable local optimal solution. Due to noise, locally optimal solution in pairwise matching may not be the correct one. Consecutively combining such local solutions, therefore, also becomes unreliable. Fig. 6(b) shows such an example: $\{(B_1, C_1), (B_3, C_2), (B_5, C_3)\}$ is a local optimum. So simply combining local solutions in a directed loop will finally result in $\{(A_3, B_5, C_3)\}$ being reported as the only final corresponding triplet, which is incorrect. In contrast, solving one global N-frame matching by extracting N-tuples from N frames simultaneously, often results in a better global solution (c).

Fig. 7 shows another example in the reconstruction of an office scene using 3-frame matching (a) scanned by a camera under relatively big shift. When pairwise 2-frame graph matching is adopted (b), even after the directed loop refinement, the reconstruction is undesirable (c). Applying a 3-frame graph matching, in contrast, results in spatially consistent 3-tuples and a desirable reconstruction (e).

Further discussion. Note that, if the common overlap region is really small and not enough consistent feature points can be found on multiple frames, then matching these many frames simultaneously will not work. In our implementation, we perform the multi-frame matching if at least $\eta_L = 10$ consistent feature points are found

Table 2
Comparison of SLAM and Reconstruction Results from Different Algorithms, in Data sets [19] with Ground Truth. The derived camera trajectory error is reported (in meters, following the unit used in [19]). Our 3-frame graph matching algorithm results in 28%, 69%, and 68% smaller error in average than [19], and also better than the SIFT+RANSAC and pairwise graph matching algorithms. The SIFT+RANSAC algorithm results in a larger error than the others, probably due to the existence of motion blur.

Methods	Data Sets (Mean Error\ RMSE)		
	fr1 desk	fr1 room	fr2 desk
SIFT+RANSAC [29,9]	0.075\0.083	0.115\0.095	0.092\0.088
SURF+RANSAC+Loop Closure [19]	0.021\0.026	0.087\0.087	0.053\0.057
Pairwise Graph Matching	0.031\0.04	0.070\0.187	0.040\0.083
Our Multi-frame Graph Matching	0.015\0.023	0.027\0.058	0.017\0.045

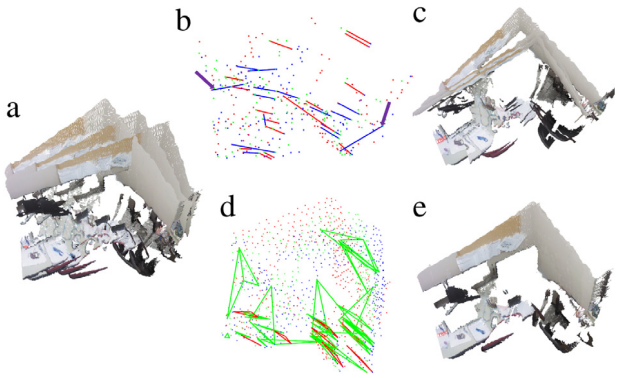

Fig. 7. (a) Three scans of an office. (b) Pairwise 2-frame graph matching identifies many corresponding pairs, among which both correct and incorrect ones exist. This results in an undesirable stitching and reconstruction (c). (d) shows the consistent 3-tuples through 3-frame graph matching, which results in a better reconstruction (e).

in all the N frames. In practice, the Kinect is generating frames rapidly. For example, on the public benchmark data sets [19], even if we down-sample the frame rate to 1/20 (about the same bandwidth our Robot can send out for processing), finding enough consistent feature points on consecutive 3 to 4 frames is usually not a problem. Meanwhile, also note that our proposed algorithm chooses suitable number of frames to match adaptively: when there are no enough common feature points, fewer number of consecutive frames will be taken for matching computation.

4. Experimental results

We evaluate our algorithm using two types of experiments. (1) 3D reconstruction on some public data sets. (2) Run our algorithm on an iRobot that navigates inside a building and performs the reconstruction.

4.1. Reconstruction on public data sets

We use our multi-frame graph matching algorithm to stitch several publicly available data sets from [19]. Our results are compared with that of the RANSAC-based algorithm, pairwise graph matching, and the ground truth data set. Table 2 shows the transformation mean error and RMSE (Root Mean Square Error) of these results. In the table, we show the results of three randomly selected data sets, "fr1 desk", "fr1 room", and "fr2 desk" in [19], comparing with our approach, the method of Endres et al. [19], SIFT + RANSAC approach (recently adopted in [29,9]), and the pairwise graph matching. Two reconstruction results are shown in Figs. 8 and 9. *Error Metrics:* The data sets from [19] are measured by an Asus Xtion Pro Live sensor. They come with ground truth that records the sensor trajectory. Therefore, we also compute the sensor trajectory using transformation matrix derived from the matching of frames. We compare the computed trajectory with the ground truth.

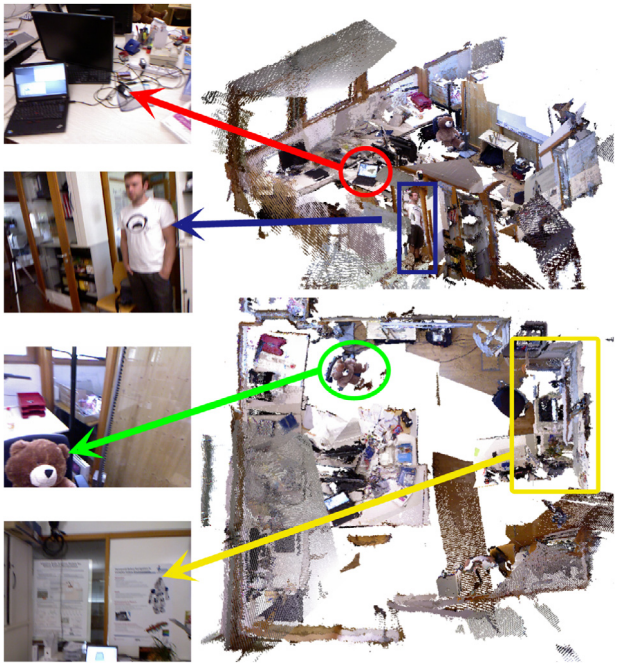

Fig. 8. Public data set of fr1 room. Details in the scanned RGB pictures are restored and constructed in the 3D scene.

Fig. 9. Public data set of fr1 desk. This scene is constructed by downsampling the frame rate to 1/20 from the original data set. The result shows that the multi-frame graph matching is robust against noise and big camera shifts.

In these experiments, to mimic the low-frame rate in the practical low-bandwidth robotic reconstruction environment like ours, we downsample the frame rate to 1/20 from the original data set. Smaller overlaps then exist between consecutive frames. Our algorithm stitches these downsampled frames and obtains the camera trajectory reliably.

In common SLAM algorithms, the *loop closure* constraint is often utilized to suppress the accumulated error and refine the alignments [49,19]. Here we can also adopt such a loop closure refinement in the end. But since this is not the focus of this project, and we want to demonstrate the effectiveness of the multi-frame matching algorithm (over the pairwise matching), here all the data we report are without any loop closure refinement.

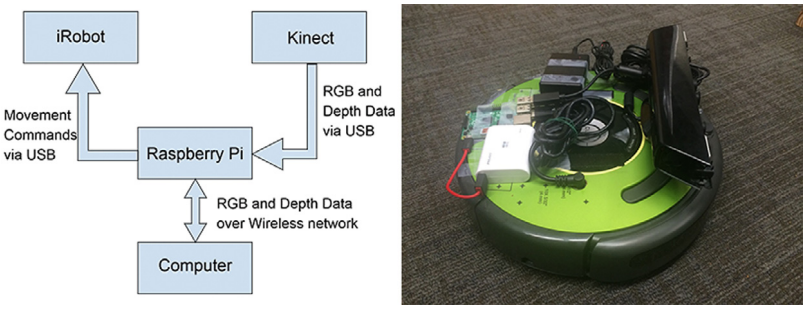

Fig. 10. (Left): Diagram of system architecture. The Raspberry Pi acts as a hub and controller for the rest of the devices in the system. (Right): Our iRobot carrying a Raspberry Pi and a Kinect.

Fig. 11. Reconstructing a printing room.

Fig. 12. Reconstructing a laboratory.

Fig. 13. Reconstructing an office room.

When solving matching for these low-bandwidth scenarios (namely, temporal data are not densely sampled and have relatively small overlap), our multi-frame matching algorithm outperforms dense matching approaches (e.g. Kinect Fusion) that utilize the ICP-based algorithms to align consecutive frame pairs. Due to ICP's sensitivity to local optima, these approaches require high frame-rate scan data to have small inter-frame transformations. In contrast, our algorithm can handle data with sparse sampling rates and bigger inter-frame transformations, because of its feature-based matching strategy. Also, compared with other pairwise feature matching based approaches, multi-frame matching exhibits better reliability, especially when the overlap is relatively small and pairwise feature correspondence could be ambiguous (see an example in Fig. 6).

4.2. Reconstruction using our iRobot system

Design of the iRobot SLAM system. Our robotic navigation system is built on an iRobot Create carrying a Microsoft Kinect 360, a Raspberry Pi I with 700 MHz single core, a USB WIFI adaptor, and two batteries for Kinect and Raspberry Pi. Remotely, the SLAM and reconstruction are done on a PC system with Intel Core i5-4440 CPU at 3.1 GHz and an 8G memory. The matching and control programs are implemented in C++. Each frame of the Kinect scan includes a 640 × 480 resolution depth image and an RGB image.

System pipeline. We remotely control the navigation of the iRobot inside each room by sending commands to the Raspberry Pi; the Raspberry Pi drives the iRobot, and simultaneously collects and transmits each frame scanned by the Kinect to the remote PC. After the transmitted data arrive, the PC starts to match them with previous frames. Fig. 10 shows the system structure and a moving iRobot carrying the Raspberry Pi and Kinect.

We use our SLAM system to scan and map three indoor environments: a printing room, a geology lab, and a working office. The 3D reconstruction results are shown in Figs. 11–13. Data from the first two experiments are collected by a navigating iRobot, while data from the third scene is done by a hand held Kinect (simply because we would like to test the reconstruction from a larger variation on the scanner's altitude and orientation). We do not implement the matching algorithm in parallel using GPUs. The running time of the matching algorithm on CPU is reported in Table 3. With about 100–200 features in each frame, the average time cost to compute a multi-frame graph matching is about 1.7 s.

Table 3
Runtime Table for our Multi-frame Graph Matching. $\overline{|P|}$ is the average size of keypoint set in each frame; $\#_F$ is the frame number; $\overline{\mathcal{L}^0}$ is the average size of initial corresponding N-tuples. T_{MGM} is the total computational time of the multi-frame graph matching algorithm (in seconds); and T_{PF} is the average stitching time per frame. The "fr1 desk", "fr1 room", and "fr2 desk" are public data sets from [19].

Scenes	$\#_F$	$\overline{\lvert P\rvert}$	$\overline{\mathcal{L}^0}$	T_{MGM}	T_{PF}
"fr1 desk"	29	78	2999	41.715	1.545
"fr1 room"	68	107	4029	171.27	2.595
"fr2 desk"	148	226	1495	66.576	0.456
"Printing Room"	35	82	3187	45.573	1.381
"Geology lab"	29	97	3513	45.522	1.686
"Office Room"	31	208	4411	72.819	2.511

5. Conclusions

We presented a novel N-frame graph matching algorithm to handle partial matching for 3D reconstruction. The algorithm is effective for aligning data sets with relatively small overlap and is robust in handling noisy data obtained by low-cost scanner

scanners such as Kinect and PrimeSense. We also develop an iRobot SLAM system to navigate and reconstruct a 3D indoor environment. The proposed N-frame graph matching model first extracts a set of N-corresponding tuples. Then the mutual spatial consistency of these features (among multiple frames), together with their descriptor similarity, is used to find an optimal correspondence among these features. Feature correspondences are then used to align and stitch frames together. Experiments show that our algorithm has better reliability than existing algorithms in the reconstruction of *noisy and low-frame-rate* 3D scans.

Despite better accuracy and robustness, a limitation of multi-frame matching is efficiency, especially when simultaneously matching many frames. Now that in 3D reconstruction, frames only differ by rigid transformations, the correct rate of the correspondence is more important than the number of corresponding pairs that are identified. Hence, increasing the similarity threshold of descriptors will suppress the size of initial correspondence set, and will greatly reduce the dimension of the affinity matrix. We also plan to explore better optimization strategies [50] and parallel implementation to improve the speed of matching computation.

Acknowledgments

This work was done when the first author was a visiting student at Louisiana State University. The work was partly supported by the National Natural Science Foundation of China [51135004], the Hi-Tech Research and Development Program of China [2012AA040701], and US National Science Foundation IIS-1320959.

References

[1] Microsoft, 2015. URL http://www.microsoft.com/en-us/kinectforwindows/.
[2] Apple, 2015. URL http://en.wikipedia.org/wiki/PrimeSense.
[3] Chou Y-S, Liu J-S. A robotic indoor 3d mapping system using a 2d laser range finder mounted on a rotating four-bar linkage of a mobile platform. Int J Adv Robot Syst 2013;10.
[4] RaspberryPiOnline, 2015. URL https://www.raspberrypi.org/.
[5] Izadi S, Kim D, Hilliges O, Molyneaux D, Newcombe R, Kohli P. et al. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera, In: ACM symposium on user interface software and technology, 2011.
[6] Whelan T, Kaess M, Fallon M, Johannsson H, Leonard J, McDonald J. Kintinuous: Spatially extended kinectfusion, 2012.
[7] Whelan T, Johannsson H, Kaess M, Leonard JJ, McDonald J. Robust real-time visual odometry for dense rgb-d mapping, In: Intl. conf. robotics and automation, ICRA, 2013. p. 5724–31.
[8] Steinbrucker F, Sturm J, Cremers D. Volumetric 3d mapping in real-time on a cpu. In: Intl. conf. robotics and automation, ICRA, 2014. p. 2021–8.
[9] Zhang K, Zheng S, Yu W-y, Li X. A depth-incorporated 2d descriptor for robust and efficient 3d environment reconstruction. In: Intl. conf. computer science & education. IEEE; 2015. p. 691–6.
[10] Nießner M, Dai A, Fisher M. Combining inertial navigation and icp for real-time 3d surface reconstruction. Eurographics 2014;13–6.
[11] Endres F, Hess J, Engelhard N, Sturm J, Cremers D, Burgard W. An evaluation of the rgb-d slam system. In: Intl. conf. robotics and automation, ICRA, 2012. p. 1691–6.
[12] Henry P, Krainin M, Herbst E, Ren X, Fox D. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: Experimental robotics. Springer; 2014. p. 477–91.
[13] Engelhard N, Endres F, Hess J, Sturm J, Burgard W. Real-time 3d visual slam with a hand-held rgb-d camera. In: Proc. RGB-D workshop on 3D perception in robotics, vol. 180. European Robotics Forum; 2011.
[14] Du H, Henry P, Ren X, Cheng M, Goldman DB, Seitz SM, Fox D. Interactive 3d modeling of indoor environments with a consumer depth camera. In: Proc. intl. conf. ubiquitous computing, 2011. p. 75–84.
[15] Kümmerle R, Grisetti G, Strasdat H, Konolige K, Burgard W. g 2 o: A general framework for graph optimization. In: Intl. conf. robotics and automation, ICRA. IEEE; 2011. p. 3607–13.
[16] Zhang K, Li X. A graph-based optimization algorithm for fragmented image reassembly. Graph. Models 2014;76(5):484–95.
[17] Zhang K, Yu W, Manhein M, Waggenspack W, Li X. 3D fragment reassembly using integrated template guidance and fracture-region matching. In: International conference on computer vision, ICCV, 2015. p. 2138–6.
[18] Li X, Iyengar SS. On computing mapping of 3d objects: A survey. ACM Comput Surv 2015;47(2):34:1–34:45.
[19] Endres F, Hess J, Sturm J, Cremers D, Burgard W. 3-d mapping with an rgb-d camera. IEEE Trans Robot 2014;30(1):177–87.
[20] Chen K, Lai YK, Hu SM. 3d indoor scene modeling from rgb-d data: a survey. Comput Vis Media 2015;1(4):1–12.
[21] Kerl C, Sturm J, Cremers D. Dense visual slam for rgb-d cameras. In: 2013 IEEE/RSJ international conference on intelligent robots and systems, IROS. IEEE; 2013. p. 2100–6.
[22] Zhou Q-Y, Koltun V. Dense scene reconstruction with points of interest. ACM Trans Graph 2013;32(4):112.
[23] Choi S, Zhou Q-Y, Koltun V. Robust reconstruction of indoor scenes. In: Computer vision and pattern recognition, CVPR. 2015. p. 5556–65.
[24] Zhang Y, Xu W, Tong Y, Zhou K. Online structure analysis for real-time indoor scene reconstruction. ACM Trans Graph 2015;34(5):1–13.
[25] Lee D, Kim H, Myung H. Gpu-based real-time rgb-d 3d slam. In: 2012 9th international conference on ubiquitous robots and ambient intelligence, URAI. IEEE; 2012. p. 46–8.
[26] Bedkowski J, Maslowski A, De Cubber G. Real time 3d localization and mapping for USAR robotic application. Ind Robot 2012;39(5):464–74.
[27] Glocker B, Izadi S, Shotton J, Criminisi A. Real-time rgb-d camera relocalization. In: 2013 IEEE international symposium on mixed and augmented reality, ISMAR. IEEE; 2013. p. 173–9.
[28] Miksik O, Mikolajczyk K. Evaluation of local detectors and descriptors for fast feature matching. In: 2012 21st international conference on pattern recognition, ICPR, 2012. p. 2681–4.
[29] Chen K, Lai Y, Wu Y-X, Martin RR, Hu S-M. Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. ACM Trans Graph 2014;33(6).
[30] Lo T-WR, Siebert JP. Local feature extraction and matching on range images: 2.5 d sift. Comput Vis Image Underst 2009;113(12):1235–50.
[31] Scovanner P, Ali S, Shah M. A 3-dimensional sift descriptor and its application to action recognition. In: Proceedings of the 15th international conference on multimedia. ACM; 2007. p. 357–60.
[32] Knopp J, Prasad M, Willems G, Timofte R, Van Gool L. Hough transform and 3d surf for robust three dimensional classification. In: Computer vision–ECCV 2010. Springer; 2010. p. 589–602.
[33] Sipiran I, Bustos B. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. Vis Comput 2011;27(11):963–76.
[34] Matei B, Shan Y, Sawhney HS, Tan Y, Kumar R, Huber D, et al. Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation. IEEE Trans Pattern Anal Mach Intell 2006;28(7):1111–26.
[35] Zhong Y. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In: Int'l conf. on computer vision workshops, 2009. p. 689–96.
[36] Steder B, Rusu RB, Konolige K, Burgard W. Narf: 3d range image features for object recognition. In: workshop on defining and solving realistic perception problems in personal robotics, int. conf. intelligent robots and systems, IROS. vol. 44, 2010.
[37] Lowe DG. Distinctive image features from scale-invariant keypoints. Int J Comput Vis 2004;60(2):91–110.
[38] Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 1981.
[39] Atkinson AC, Riani M. The forward search and data visualisation. Comput Statist 2004;19(1):29–54.
[40] Leordeanu M, Hebert M. A spectral technique for correspondence problems using pairwise constraints. In: Intl. conf. computer vision, ICCV. vol. 2. IEEE; 2005. p. 1482–9.
[41] Torresani L, Kolmogorov V, Rother C. Feature correspondence via graph matching: Models and global optimization. In: Computer vision–ECCV 2008. Springer; 2008. p. 596–609.
[42] Cho M, Lee J, Lee KM. Reweighted random walks for graph matching. In: Computer vision–ECCV. Springer; 2010. p. 492–505.
[43] Cheng Z-Q, Chen Y, Martin RR, Lai Y-K, Wang A. Supermatching: feature matching using supersymmetric geometric constraints. IEEE Trans Vis Comput Graphics 2013;19(11):1885–94.
[44] Olivier D, F B, Jean P. A tensor-based algorithm for high-order graph matching. IEEE Trans Softw Eng 2011;33(12):1980–7.
[45] Leng C, Xu W, Cheng I, Basu A. Graph matching based on stochastic perturbation. EEE Trans Image Process 2015;24(12):1.
[46] Hänsch R, Webera T, Hellwicha O. Comparison of 3d interest point detectors and descriptors for point cloud fusion. ISPRS Annals of the Photogrammetry. Remote Sens Spat Inf Sci 2014;57–64.
[47] Eggert DW, Lorusso A, Fisher RB. Estimating 3-d rigid body transformations: a comparison of four major algorithms. Mach Vis Appl 1997;9(5–6):272–90.
[48] Knupp PM. Algebraic mesh quality metrics for unstructured initial meshes. Finite Elem Anal Des 2003;39(3):217–41.
[49] Newman P, Ho K. Slam-loop closing with visually salient features. In: IEEE intl. conf. robotics & automation, 2005. p. 635–42.
[50] Park S, Davis TA, Hager WW, Zhang H. Quadratic programming techniques for graph partitioning, 2006.