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/© 2016 Elsevier Ltd. All rights reserved.

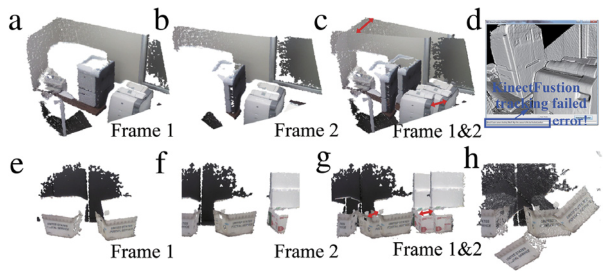

Fig. 1. Kinect-fusion (a–d) and ICP-based (e–h) algorithms fail to stitch frames when camera shift is big (i.e., the overlap between frames is small). (a–b) are the two frames to match, (c) is a global view of the offset between the two frames in a same coordinate, and (d) shows a snapshot where the 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].