Scalable Multi-Device SLAM

Jack Morrison, Dorian Gálvez-López, and Gabe Sibley
School of Engineering and Applied Science
George Washington University
Washington, DC 20052
{jackmorrison, dorian, gsibley}@gwu.edu

***Abstract*—This paper is concerned with enabling distributed 3D mapping and map sharing. The proposed approach allows users with mobile devices to collectively construct, stream and share a scalable model of the world. The resulting maps are accurate and allow for precise six-degree-of-freedom localization. Particular effort is paid to ensure that the system is scalable in both spatial extent and number of simultaneous users. Using a relative manifold provides a unified representation that is both amenable to simultaneous asynchronous access and easily scales with new data. This enables multi-session map building and concurrent map fusion by detecting cross loop closures. Maps are built to be invariant to the sensing camera to support heterogeneous sensors. This paper demonstrates this capability on cellphone video and IMU data, showing crowdsourced mapping from multiple devices.**

I. Overview

Present day cellphones are ubiquitous, computationally powerful and have increasingly high-quality sensors and displays. Many of the devices on the market are already capable of significant computation and, of course, are frequently connected to the Internet, allowing for global communications. With these devices, there is a great opportunity to make crowdsourced 3D mapping a reality.

Mapping the world with mobile devices puts mapping capabilities in the hands of millions of existing cellphone users. With on-device mapping, a cellphone can become an integrated part of the physical world. This, however, will require a mapping solution which can enable multiple devices to simultaneously query and update a global map.

Existing SLAM solutions do not scale well enough to withstand crowdsource size maps. Previous multi-device or multi-session sub-mapping solutions solve the problem by connecting multiple locally independent and metric maps, but this can cause problems when local sub-maps overlap and requires the system to choose an "active" sub-map to track. Other approaches require expensive fusion operations when joining maps, due to their privileged frame representations. Both of these approaches have scaling limitations.

This paper presents a system for simultaneously building and sharing large-scale 3D maps from multiple monocular mobile devices. It describes infrastructure, algorithms and core data structures for building and sharing arbitrarily scalable visual maps.

The presented system operates in a client-server architecture, receiving and distributing maps from servers that provide endpoints for map related queries. By operating behind a simple API, the system allows clients to easily integrate server queries into their processing when a network connection is available. The server API is also purely image driven, removing the need for an accurate global location when interacting with the server.

II. Related Work

Previous approaches to multi-device mapping fall largely into two camps: "hybrid" sub-mapping techniques and privileged frame fusion approaches. In fully fixed frame representations, like [14], an expensive map merging algorithm is required to join measurements and objects in multiple maps together. These approaches offer simplicity in representation, but would not scale well because of the need to reprocess all data before merging maps. The sub-mapping techniques are based on limited-area metric maps, such as the occupancy grids used in [4] or the planar AR workspaces employed in the multiple map extension to PTAM [7], PTAMM [3]. These approaches apply a priviliged frame SLAM algorithm to populate their maps, but in order to bound complexity, after a map's extent has grown too large they initialize a new and fully independent metric map. In PTAMM, they reuse their existing relocalization techniques to switch the system's focus between maps, and in both [4, 10], they connect maps using topological linkages based on odometry or "anchor nodes". These approaches recognize the bounds of privileged frame representations, but still maintain a dependency on them. In their conclusion Castle et al. [3] discuss future work opportunities that this paper deals with, including IMU integration and the fusion of many sequences.

Forster et al. [5] and Riazuelo et al. [12] present a multi-threaded method for building maps using Micro Aerial Vehicles (MAVs) and mobile robots. Their approaches both seek to decouple the motion estimation and map building pipelines by communicating keyframe information to a central station where it is fused into a cohesive map structure. Loop closure events and map construction are handled on this central station while the client is left to perform visual odometry with only the map it is provided by the server. These solutions lower the computational cost on the client, but can reduce its autonomy in the face of itermittent networking. In contrast, our system allows clients to create individual maps even if connection to the server is lost. Individual maps are fused by cross loop closures yielded by the server, as done by McDonald et al. [9]. However, unlike them, we do not require to extract new

features to perform this operation since we exploit the features tracked by the front-end. In addition, we reuse the map graph to check for loop consistency.

III. System Description

A. Architecture

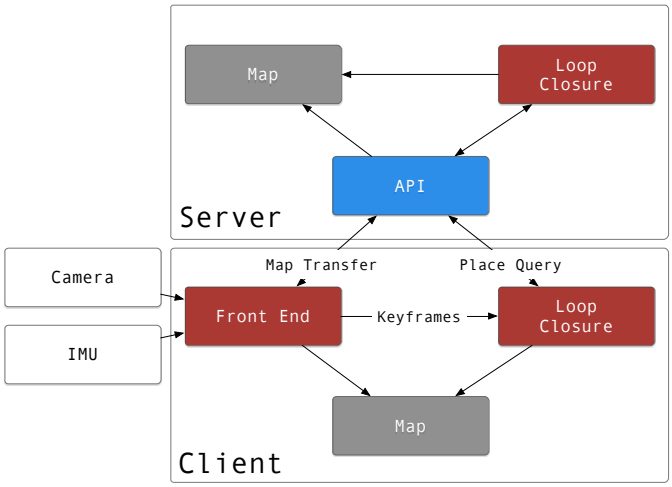

Fig. 1. System Architecture Outline

The system described here is divided into a client-server model, as shown in Figure 1. The client has all the capabilities for SLAM, including map-building and loop closure, while the server is used as a repository for storing maps from multiple sessions. This design allows the client to operate completely on its own when a network connection is unavailable.

The main components of the client, which run on separate threads, are the front-end and the loop closure subsystem. The front-end is responsible for the bulk of the SLAM pipeline including image processing, feature tracking, visual-inertial pose and landmark estimation, and map building. The loop closure back-end receives notifications of new keyframes from the front-end thread and is responsible for detecting loops in the client's trajectory. The loop closure system is tightly integrated with the map generated by the front-end, reusing its measurement points on keyframes. Both threads interact asynchronously with the server through its API. The server system is responsible for large-scale storage of maps and for answering loop-closure style place queries across its entire database of SLAM sessions.

At the heart of this system is its relative map data structure. The map is a relative pose-graph with metric landmark information stored in its node of the graph. The relative framework is key to this system's scalability and ease of use. The same map system is used by both client and server. It allows for persistent map storage and bounded RAM usage to ensure constant resource utilization for long-term performance.

B. Relative Manifold

The key to the approach presented here is the single, seamless map structure that can be shared across multiple devices.

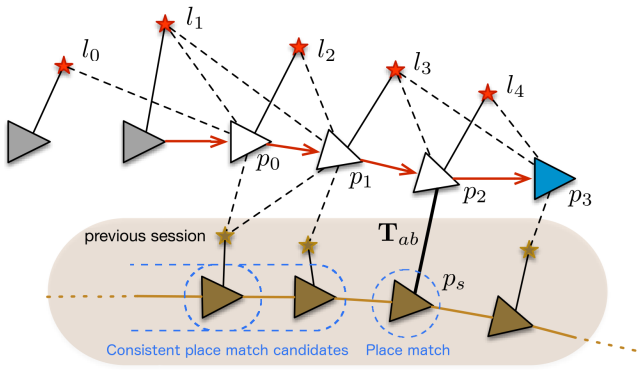

Fig. 2. Graphical representation of the constraints after loop closure.

The map is represented as in RSLAM [11]: by an undirected graph of nodes, representing key frames, connected by transformation edges which encode a pose estimate between a pair of frames. The frames carry landmarks and measurements of landmarks. To allow the reuse of patches identified in the cameras of other devices, the camera calibration used for each session is also stored.

Each of the objects in the system have an identification structure which refers back to the map building session during which they were created. Every session receives a UUID which is used to label frames, edges, measurements, and landmarks created as a part of that session. Additionally, the camera calibration used during each session is associated with a session UUID for later retrieval. The identification hierarchy provides uniqueness of sessions, frames, measurements, and landmark references during processing and transfer. In this way, the server and client can be confident that they are referring to the same data.

Unlike single-frame representations, the relative graph structure of RSLAM allows sections of mapped environments to be referenced independent of other mapped locations. There is no need to transmit the entire graph or transform nodes before interacting with them. The ability to interact and independently alter sections of the map is important for many operations, including loop closures and multi-session mapping. This is also key to making downloaded sections of map available immediately. The system does not depend on knowledge of a global coordinate frame to make use of frames, so pieces of the global map may be downloaded as they are made available, or as bandwidth allows.

C. Distributed Mapping

To distribute and aggregate maps, our system is designed in a client-server architecture. The server is a centralized repository for map information, including (possibly disjoint) maps, camera calibrations, and place matching (loop closure) information. The server's role is to act as a hub for distributing this information to connected clients.

The server API has three request types: place queries, downloads, and uploads.

1) Place Query: The client uploads an image to the server to be matched against the server's database of places. Each place on the server is correlated with a key frame, so when a match is made, that key frame is loaded and the relative position of the query image is estimated. A new edge that connects the query frame and the matched frame is returned to the client. Through this edge, the local BFS operation can access and include landmarks from the downloaded session and use them in localization.

2) Map Download: The client can request a section of the graph by specifying a frame ID and a depth to which it would like a breadth-first search to be performed. This search will gather frames, edges, camera calibrations, and all associated metadata, and return it to the requesting client. The result also includes "leaves" of the graph, nodes which were not downloaded but are at the edge of the fetched map. This gives the client locations to possibly request more map, if it requires it.

3) Map Upload: The client serializes frames, edges and associated places and send them to the server for later querying by other devices. These are inserted into the server's databases, which fuses them automatically if there are any conjoining edges.

D. Loop Closure

For every keyframe created during a mapping session, the loop closure system tries to detect places that have been already visited by this or any other session, so that it may close a loop or join two disconnected maps. The place matching is fully integrated in the system and takes advantage of the information available on the clients and on the server. Each of them integrates the loop detector by Gálvez-López and Tardós [6] into its map graph and operate in the same way. They store a database to describe places as bags of binary words by using a single hierarchical vocabulary comprising 10^5 words, trained offline with millions of features obtained from independent data.

The client computes ORB [13] descriptors around the features tracked by the front-end. This provides a low number of points that are highly repeatable, are well distributed around the image and are associated to landmarks with estimated 3D poses. In addition, we avoid the overhead of computing new features by reusing the map's key points. These descriptors are converted into a bag-of-words vector and the database is queried. The top-100 candidate matches are grouped together if their reference frames are close in the map, and the group with the highest aggregated similarity score is kept. If the system runs with a stereo camera or an IMU and the map is then scaled, we measure distances between frames in the Euclidean space, otherwise, we compute the geodesic distance. To avoid false detections, the candidates are accumulated until subsequent queries are resolved. When at least 3 of them are close each other, we try to close the loop. For that, by comparing ORB descriptors, we obtain 2D-to-3D correspondences between the current image and the matched 3D landmarks. Solving the perspective-*n*-problem yields a relative transformation that is later optimized by including other measurements.

If a transformation can be found through PnP, the place match is accepted and an edge is added to the map between the querying frame and the matched frame. Through the relative manifold structure, all landmarks on the matched frame and nearby frames are now available to the front-end for localization, including any frames which were downloaded from a server. This $O(1)$ operation is the entirety of "map fusion" in this system and is the main drive behind its efficiency.

The server operates as described above, but manages a larger joint map that comprises the places visited by each client. Thus, after consecutive *place query* requests are consistently satisfied, an edge is added between two different session maps, as shown in Figure 2.

IV. Results

A. Performance of Feature Tracks for Loop Detection

TABLE I
PERFORMANCE OF FEATURES FOR LOOP DETECTION

Features	Computation time (ms/image)	Number of detections
ORB 50	7.0	66
ORB 100	7.8	105
ORB 300	8.7	115
Tracks (≈ 52)	1.4	107

To demonstrate the performance of using the tracked features in the loop detection stage, the system was run on a dataset containing 1639 stereo images in which the camera describes a figure-eight-shaped trajectory that is traversed three times, yielding around 200 revisited places. The front-end feature tracks were compared with the features yielded by the ORB keypoint detector from OpenCV [2], which does not take into account the distribution of the points in the image, or their persistence across frames.

Table I shows the time required to compute the features and the number of loops detected with them. The average number of tracks in this sequence was 52 per image. By using tracked features extra keypoint extraction time is avoided without any loss of performance in the loop detector.

B. Multi-session Map Joining

We processed three short urban trajectories using the client-server system and overlay their combined paths on a map in Figure 3. We recorded IMU and images on a modern mobile device and then replayed the data on a laptop to capture system behavior. The figure shows the potential for streaming map data to the cloud and back to devices for reuse. This small image displays approximately 900 m of device trajectory, which is built by transmitting it in segments, as updates are desired.

V. Discussion

Our proposal presents the first steps towards an architecture for long-term SLAM that addresses scalability. There are open avenues for improvement, however. For instance, our

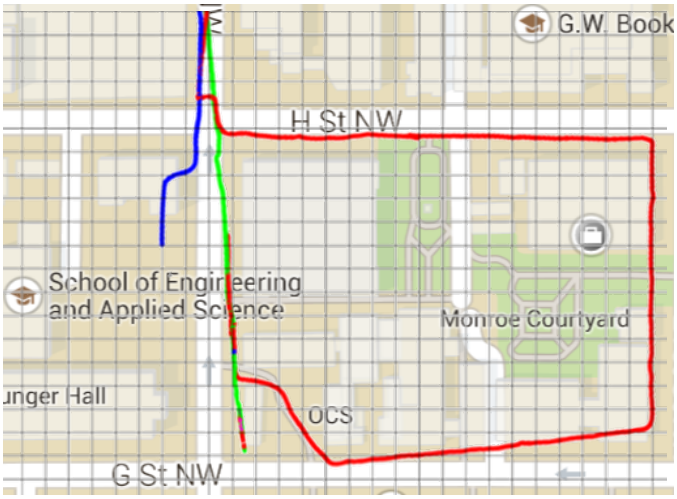

Fig. 3. Urban Multi-Session Map

loop detection method is based on a single visual vocabulary. Previous research has shown that this approach is suitable for heterogeneous environments mapped with very different cameras [6]. Our next step is to research the limits of this approach when mapping trajectories of hundreds of kilometers. On the other hand, no algorithm is completely exonerated from false positives under all circumstances. Thus, long-term robustness can be achieved by applying a recent technique such as *Realizing, Reversing, and Recovering* (RRR) [8], which accumulates several loop hypotheses to remove later those that are inconsistent. Inconsistencies in uploaded maps have also not been addressed yet, but may be addressed by applying a consensus policy [1] to reject bad maps.

The independence of subsections of the map structure presented here makes horizontal scaling a potentially valuable future work direction. Currently the system is based around a single server, but extending the system to distribute map endpoints should be straight forward thanks to the underlying structures and the globally unique identifiers. Additionally, more work is planned on the server-side curation of maps, including large-scale bundle adjustments, additional inter-session joins and map sparsification. These additions could improve map accuracy and compactness.

To conclude, this paper presented a new client-server based system for joining maps created from mobile devices. The mobile client runs a full SLAM pipeline to allow for autonomous operation and the system's server component is responsible for large-scale map storage and map distribution. This system is based on a relative map structure and globally unique identification tags which allow scalable growth. The results presented in this paper came from relatively short selections of data, but qualitative results are promising and larger scale experiments are planned.

References

[1] R. Aragues, J. Cortes, and C. Sagues. Distributed consensus on robot networks for dynamically merging feature-based maps. *IEEE Transactions on Robotics*, 28 (4):840–854, Aug 2012.

[2] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Robert O. Castle, Georg Klein, and David W. Murray. Wide-area augmented reality using camera tracking and mapping in multiple regions. *Computer Vision and Image Understanding*, 115(6):854–867, June 2011.

[4] H.J. Chang, C. S G Lee, Y.C. Hu, and Yung-Hsiang Lu. Multi-robot SLAM with topological/metric maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007*, pages 1467–1472, 2007.

[5] Christian Forster, Simon Lynen, Laurent Kneip, and Davide Scaramuzza. Collaborative monocular SLAM with multiple micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3962–3970. IEEE, 2013.

[6] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[7] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[8] Yasir Latif, César Cadena, and José Neira. Robust loop closing over time for pose graph SLAM. *The International Journal of Robotics Research*, 32(14):1611–1626, 2013.

[9] J McDonald, M Kaess, C Cadena, J Neira, and JJ Leonard. Real-time 6-DOF multi-session visual SLAM over large-scale environments. *Robotics and Autonomous Systems*, 61(10):1144–1158, 2013.

[10] John McDonald, Michael Kaess, Cesar Cadena, Jos Neira, and John J Leonard. 6-DOF multi-session visual SLAM using anchor nodes. In *European Conference on Mobile Robotics, Orbero, Sweden*, volume 13, 2011.

[11] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. RSLAM: a system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94(2):198–214, September 2011.

[12] L. Riazuelo, Javier Civera, and J.M.M. Montiel. C2TAM: a cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 62(4):401–413, April 2014.

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011.

[14] N. Ergin zkucur and H. Levent Akn. Cooperative multi-robot map merging using fast-SLAM. In Jacky Baltes, Michail G. Lagoudakis, Tadashi Naruse, and Saeed Shiry Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, number 5949 in Lecture Notes in Computer Science, pages 449–460. Springer Berlin Heidelberg, January 2010.