2015 IEEE International Symposium on Mixed and Augmented Reality

[POSTER] Tracking and Mapping with a Swarm of Heterogeneous Clients

Philipp Fleck*
Graz University of Technology

Clemens Arth†
Graz University of Technology

Christian Pirchheim‡
Graz University of Technology

Dieter Schmalstieg§
Graz University of Technology

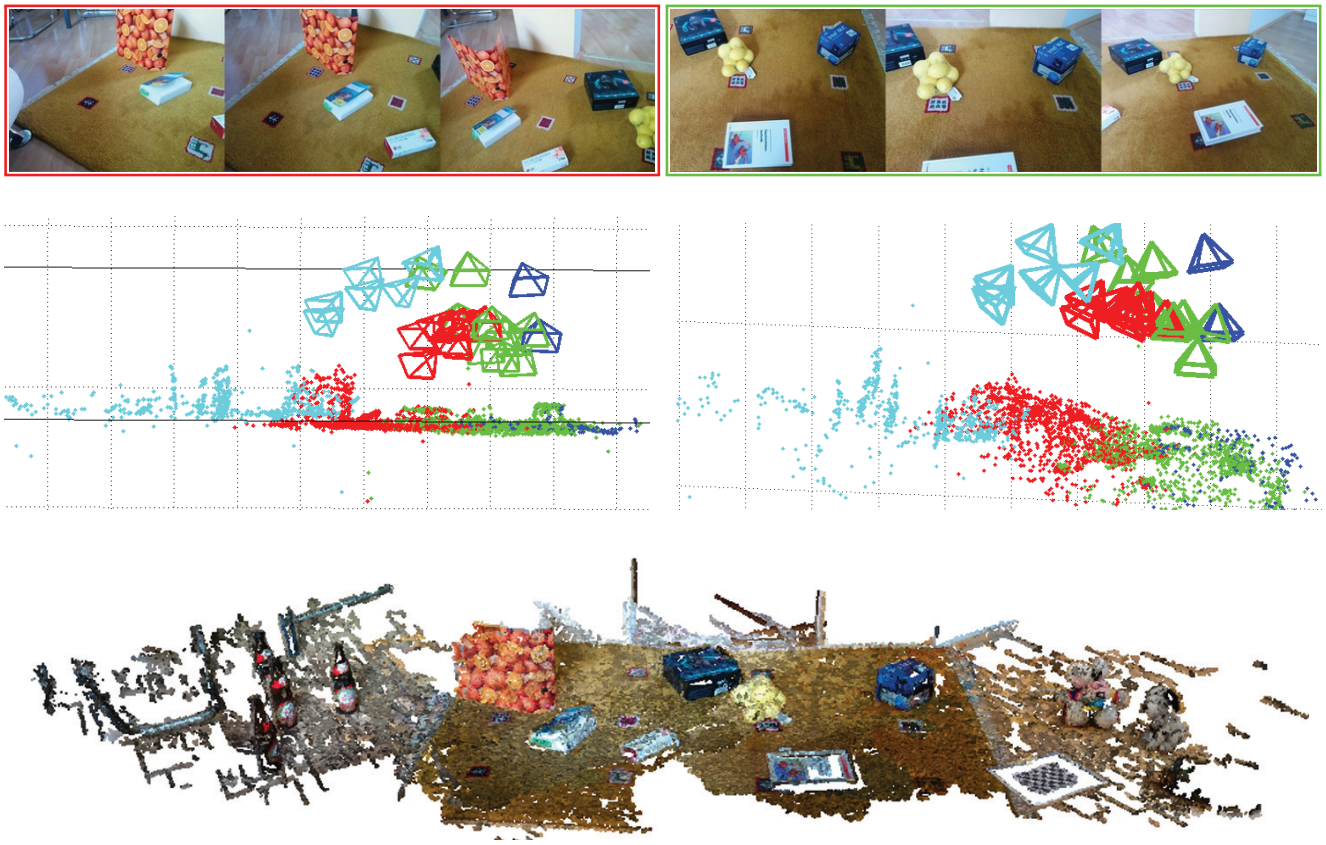

Figure 1: Collaborative Tracking and Mapping. **Top:** Keyframes of individual SLAM clients observing the same scene simultaneously. **Middle:** Sparse point map created by the server using the keyframes from four clients. **Bottom:** Densified server point cloud reconstruction of the scene.

Abstract

In this work, we propose a multi-user system for tracking and mapping, which accommodates mobile clients with different capabilities, mediated by a server capable of providing real-time structure from motion. Clients share their observations of the scene according to their individual capabilities. This can involve only keyframe tracking, but also mapping and map densification, if more computational resources are available. Our contribution is a system architecture that lets heterogeneous clients contribute to a collaborative mapping effort, without prescribing fixed capabilities for the client devices. We investigate the implications that the clients' capabilities have on the collaborative reconstruction effort and its use for AR applications.

*e-mail: philipp.fleck@icg.tugraz.at
†e-mail:arth@icg.tugraz.at
‡e-mail:pirchheim@icg.tugraz.at
§e-mail:schmalstieg@tugraz.at

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented and virtual realities; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking; J.7 [Computer Applications]: Computers in Other Systems—Real time

1 Introduction

When every person in the room has a mobile phone in their pocket, the opportunity of turning these devices into a "swarm" of smart cameras presents itself. Together, multiple mobile devices can work more efficiently on *simultaneous localization and mapping* (SLAM) than any single mobile device alone could. With tracking and mapping in a single workspace, collaborative applications for augmented reality (AR) are enabled, such as interior design or games involving physical objects.

However, a swarm of unmodified mobile devices lacks many properties that traditional multi-camera systems employed in robotics have: The devices do not have a fixed geometric arrangement. Their capabilities in terms of image quality and computational performance may differ dramatically, and even their availability may change, as users leave the room or run out of battery power.

978-1-4673-7660-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ISMAR.2015.40

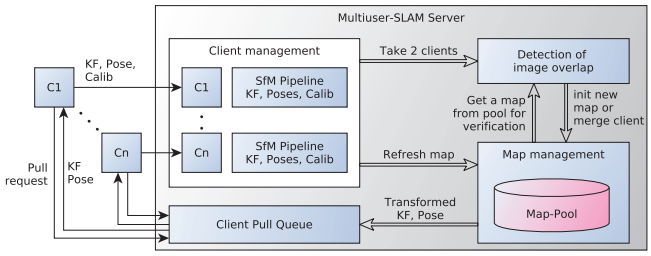

Figure 2: Client-server system architecture. The nodes $C_1 \dots C_n$ represent SLAM clients, submitting information (*e.g.* keyframes) to the SfM server. For each client, the server creates a *client management* module, where a new map is reconstructed and stored in the *map pool*. The server continuously updates the client maps and attempts to merge multiple maps. It pushes relevant information (*e.g.* keyframes with camera poses) into the *client pull queues*, from where they can be pulled by clients to update their own local map.

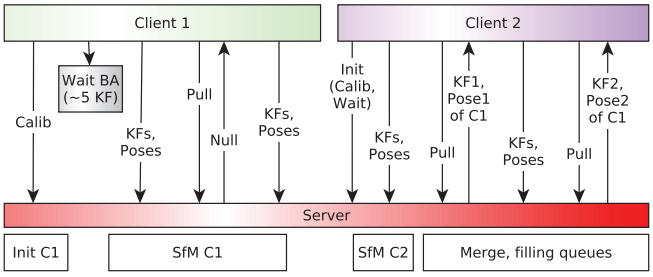

Figure 3: Server-client communication. Left: Client 1 commits its calibration, keyframes and poses, and triggers the server to start the reconstruction, while trying to pull new data. Right: Client 2 performs the same initial steps as client 1. The server starts searching for correspondences between client 1 and 2. It is able to merge the 3D structures and fills the client's queues. Pull request are now answered with keyframes and poses from the common map.

With wireless networking, the required computational performance and persistent storage can be provided via cloud computing. Conceptually, clients operate as a swarm, but, physically, the clients only connect to the cloud server. The cloud server treats keyframes obtained from the clients as an image database and computes *structure from motion* (SfM) on the database.

Unfortunately, such an attempt at using distributed computation to better amortize the computational demands of multi-user SLAM cannot operate in a tight loop between client and server. A client relying on low-latency response from the server, for example, for interest point triangulation, will fail if server or network connection are slowed down. Such a case is very likely in real-world cloud computing. Yet, it has been mostly ignored by previous multi-camera SLAM systems. In this work, we demonstrate two advances over such previous work.

First, we demonstrate that with a careful subdivision of labor, independent and heterogeneous clients are possible. We only allow clients that can operate independently, without the help of a server, for some time, to be swarm members. Members provide observations to the server, and, in return, receive updates on the other member of the swarm and on the mapped environment. This independence is achieved by executing an independent SLAM instance on every client. Beyond the basic SLAM capability, more capable clients can add additional features, such as map densification or AR applications.

Second, we demonstrated that maps that start out independently can actually be merged later, eventually leading to globally referenced maps. The server attempts to merge client maps whenever possible and thus can provide a common reference coordinate system that allows for exchanging spatially registered information between clients. This behavior has been speculatively mentioned as a possibility by some authors [10, 5], but never implemented in any published system.

We show that this approach is a robust framework that allows for collaboratively mapping and tracking a three-dimensional scene in real-time between a server and multiple mobile clients. We demonstrate the plausibility of our approach with a collaborative AR application prototype that shares augmentations across clients.

2 Related Work

For brevity, we only review work directly related to SLAM, putting special focus on approaches considering multiple users. The interested reader is referred to the relevant literature for an introduction to SfM [4].

Davison et al. [2] first introduced real-time monocular SLAM with a sparse feature map. Later, Klein and Drummond [7] proposed *parallel tracking and mapping* (PTAM), with the mapping part operating asynchronously on selected keyframes rather than in lock-step with the tracking. Those two approaches still serve as the foundations for improving, extending and studying multi-user SLAM.

Castle et al. [1] proposed an extension to PTAM handling multiple maps. The main motivation was better recovery from tracking failures, rather than multi-user mapping. Riazuelo *et al*. [9] proposed a cloud-based framework as an evolution of PTAMM. Their C^2TAM system introduces a server-client architecture, placing the mapping task entirely on the server, and leaving clients to perform only tracking. Apart from the high network bandwidth requirements of this approach, clients cannot operate without the help of the server and are, thus, not suitable for operation in arbitrary wireless networks.

CoSLAM [12] uses multiple cameras to collaboratively create a map and to handle dynamic objects. The system was demonstrated with multiple clients and a central server. However, it purely focused on reconstruction from multiple cameras and did not deal with communication issues at all.

Ventura *et al*. [11] combined a server-side localization system with a local SLAM client operating on a mobile device. Clients commit keyframes to a server which performs localization based on a pre-made and geo-registered model of the environment. The server is only responsible for localization, and its map remains static throughout the system operation. Multiple clients were not considered.

In contrast to previous work, our system only shares keyframes and scene structure, neglecting the underlying feature representation. This introduces some level of redundant computation, but allows us to integrate SfM and SLAM components into a single system. Since every node – both clients and server – can execute independently, we achieve a new level of sustainability of operation.

3 Method

Our system consist of a server running an SfM pipeline and multiple clients running SLAM. The reconstructions created by clients and server (i) use different feature descriptions, (ii) reside in different coordinate systems, and (iii) are created asynchronously, using per-node strategies involving global or local optimization.

Clients and server communicate over the network using a protocol focusing on keyframes and camera poses. Fig. 3 shows an example. After connecting to the server, a client first registers its ID and provides its internal camera calibration. The server initializes a per-client message queue. After initializing its local SLAM map, the client submits the corresponding stereo keyframe pair to the server, which reconstructs a per-client map independently of the client's own map (Section 3.1).

Figure 4: Densified individual 3D reconstructions of four clients, see also Figure 1.

During operation, the client asynchronously pulls messages from its queue (*e.g.* during idle times of background mapping thread). Upon certain events, the server puts relevant information into the client's queue. For example, if a second client transmits a keyframe that allows the merging of two clients' maps, the server offers additional keyframes and corresponding poses. The client may incorporate these keyframes into its local map (Section 3.2). Additionally, the server may provide anchor points that allow for synchronizing the reference coordinate systems between multiple client instances (Section 3.3).

3.1 Server

The server is based on the SfM pipeline described by Hoppe *et al.* [6]. It uses SIFT features [8] calculated on the GPU[1]. Every client is assigned a separate instance of the reconstruction pipeline at the server. Upon commitment of keyframes, the server creates a sparse 3D reconstruction of the scene for each client and investigates a potential image overlap between the views of clients by feature matching and epipolar geometry estimation. Based on this overlap detection, either a new map is introduced (added to the map pool) or an existing one is enlarged through merging. Maps in the pool keep pace with the client reconstructions and are refreshed through the insertion of new keyframes, as client reconstructions grow. Managing stored maps includes inserting yet unseen keyframes from other clients and the corresponding pose information into client queues (Fig. 2).

The map merging process is based on the feature point correspondences established during overlap detection. We first use the P3P algorithm [3] on a keyframe of client A with a pose $\mathbf{P}_A$ and the map of client B, thereby recovering the pose $\mathbf{P}_A^B$ of the keyframe of client A in the coordinate system of client B. A single 3D point $\mathbf{X}$ triangulated in both maps of clients A and B suffices to estimate the remaining scale factor s through Eqn. 1[2],

$$s = \frac{||\mathbf{X}_A - \mathbf{c}_A||}{||\mathbf{X}_B - \mathbf{c}_B||} \tag{1}$$

where $\mathbf{c}$ denotes the corresponding camera centers. However, we must ensure robustness, while the maps are enlarged over time. The robustness is achieved by continuously re-estimate the scale factor as the median over the distance ratios between the camera centers and all 3D points commonly triangulated.

The 3D structure from client A is transformed into the coordinate system of client B through Eqn. 2,

$$\mathbf{X}^B = (\mathbf{P}_A^B)^{-1} \cdot \begin{pmatrix} \mathbf{X'}_A \cdot s \\ 1 \end{pmatrix} \qquad \text{with} \qquad \mathbf{X'}_A = \mathbf{P}_A \cdot \begin{pmatrix} \mathbf{X}_A \\ 1 \end{pmatrix} \tag{2}$$

and the poses of the keyframes $i = 1 \ldots n$ of client A are transformed according to Eqn. 3,

$$\mathbf{P}_{(i)A}^B = \mathbf{P}_{(i)A} * \begin{bmatrix} (\mathbf{P}_A)^{-1} & \\ \mathbf{0}^T & 1 \end{bmatrix} * \begin{bmatrix} \mathbf{P}_A^B & \\ \mathbf{0}^T & s \end{bmatrix} \tag{3}$$

[1] `http://cs.unc.edu/~ccwu/siftgpu/`

[2] Poses $\mathbf{P}$ are 3×4 matrices, and 3D points $\mathbf{X}$ are 3×1 vectors.

Once a client adds a new keyframe to its local map, the corresponding server-side reconstruction is enlarged, running bundle adjustment on the newer subset of keyframes and points and fixing previous ones to maintain real-time performance, as maps grow large. Subsequently, the server provides the keyframe to all other clients observing the same scene. Each keyframe is warped to fit the client's camera calibration before being placed in the outbound queue. The server also provides the corresponding pose, transformed in the respective individual client's coordinate system, thereby avoiding additional computational effort on the client side. Finally, the server also offers 3D points and their corresponding observations, if they have been proven stable by bundle adjustment.

3.2 Clients

Various client types can become part of the swarm. The underlying SLAM system uses corners tracked across frames to create 3D map points, which differ from those built by the server. A client decides independently on the type and amount of data it wants to read from the queue provided by the server, for instance, when it has spare computational resources. We experimented with three different client configurations:

V1 The client is only reading keyframes and adding it to the map through P3P. The required feature correspondences are established through exhaustive matching of patches around corners in the new keyframe and the existing keyframes. This approach is simple, but expensive, and can occasionally lead to wrong pose estimates.

V2 The client is reading keyframes with poses from the server. These keyframes can be directly added to the local map, by extracting observations from existing 3D points through back-projection. This approach is very fast, but does not create any new 3D points based on features from server keyframes.

V3 The client improves upon **V2** by creating additional 3D points from the new keyframes. The search for matching features is guided by poses provided by the server. The additional map densification requires more computational resources than **V2**, however, it enables the client to grow it maps into yet unseen areas through keyframes from other clients.

3.3 Client-server synchronization

Our system enforces *eventual consistency* between the server and its clients over time. We synchronize corresponding client and server maps by applying the *anchor point* method of Ventura *et al.* [11]. For each remote client map, the server determines a set of well-converged 3D map points that can be used to align the corresponding local client map. These anchor points can be integrated into the local client maps as fixed points and provide strong constraints in the clients' bundle adjustment optimization. In particular, a consistent reference coordinate system is established, which is essential for collaborative AR applications, where multiple clients render virtual objects in a consistent way.

4 Experimental Results and Discussion

In this section, we present results obtained with the prototype implementation of our system on table-sized indoor scenes. Our server runs on a notebook computer and is connected with mobile device

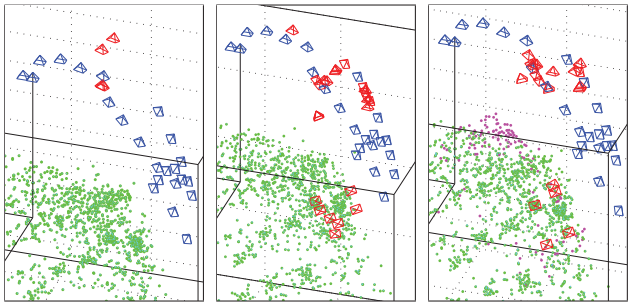

Figure 5: **Left to right:** Client mode **V1** (keyframes only), **V2** (keyframes and server poses) and **V3** (keyframes, server poses and 3D point triangulation). Blue: local poses. Red: received server poses. Green: local 3D points. Magenta: added 3D points. **V1** only added some of the received poses, because the RANSAC parametrization is set to avoid false positives. **V2** added all received keyframes and poses, and **V3** additionally triangulated new 3D points.

clients (Google Nexus 5, Samsung Tablet, Microsoft Surface Pro) via WiFi.

Figure 3 outlines the communication between the server and two clients for the reconstruction of the scene depicted in Figure 1. Network communication operates asynchronously through per-client queues, but we are not yet fully exploiting all system features. We plan to replace the individual queues with a publish-subscribe pattern that reduces network traffic

Figure 1 shows keyframes received from clients and the successfully merged map from four clients. Figure 4 shows the individual remote client maps reconstructed by the server. On rare occasions, merging problems are caused by the fact that the server employs SIFT features for matching. SIFT is not sufficiently invariant against strong tilt changes, *e.g.* when two clients observe a table from opposite sides. We also experienced occasional congestion of the client after a map merges triggered a large map update, consisting of potentially many new keyframes. This problem can be overcome by restricting the maximum rate at which a client processes map updates.

The map information provided by the server can be employed by the clients to extend and refine their local maps. We suggested three types of clients (Fig. 5). In the following, we are discussing the advantages and disadvantages of the three client types. Our baseline **V1** is obviously outperformed by the other types. The strength of **V2** is that it improves the tracking performance with low computational effort. However, no local map extension is done. **V3** represents a trade-off between tracking performance and map enhancements. We found that a good compromise is to actively switch between modes **V2** and **V3**, enabling the client to improve the tracking performance with low computational overhead and to enlarge its map whenever computationally possible.

In addition to reconstructing and maintaining a shared map pool, our server allows to register annotations, and, thus, enables simple collaborative AR applications. In our application prototype, we let mobile clients engage in a card game. Figure 6 depicts the live AR view of two clients, including shared annotations, *i.e.* the arrows pointing to one of the cards. Created by either of the clients, an annotation is first registered in the local client map coordinate system. After merging two clients maps, the server automatically transforms all registered annotations into the common reference coordinate system and pushes them to the corresponding clients. Due to the server-side calculation, no computational overhead is generated on the client-side.

5 Conclusion

We presented a heterogeneous system for tracking and mapping, consisting of a server and a swarm of mobile clients. The swarm members are largely independent, but complementary. Our prototype enables collaborative AR applications suited for mobile devices covering a wide range of performance characteristics.

Figure 6: Collaborative AR application with two clients C1, C2. **Top:** C1 annotates the yellow-blue card and commits the annotation to the server. **Bottom:** As soon as C2 pulls from the server, the arrow annotation is also visualized.

Acknowledgements

This work was partially funded by the Christian Doppler Laboratory on Handheld Augmented Reality.

References

[1] R. O. Castle, G. Klein, and D. W. Murray. Video-rate Localization in Multiple Maps for Wearable Augmented Reality. In *ISWC*, pages 15–22, 2008.

[2] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *ICCV*, pages 1403–1410, 2003.

[3] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[5] A. Hook, P. Fite-Georgel, M. Meisnieks, A. Maes, M. Gardeya, and L. Naimark. Generation and sharing coordinate system between users on mobile, 2014. US Patent: 20140267234 A1.

[6] C. Hoppe, M. Klopschitz, M. Rumpler, A. Wendel, S. Kluckner, H. Bischof, and G. Reitmayr. Online feedback for structure-from-motion image acquisition. In *BMVC*, pages 70.1–70.12, 2012.

[7] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, Nara, Japan, November 2007.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, Nov. 2004.

[9] L. Riazuelo, J. Civera, and J. Montiel. C2TAM: A Cloud framework for cooperative tracking and mapping. In *Robotics and Autonomous Systems, Volume 62, Issue 4, Pages 401-413, April 2014*, 2014.

[10] C. Sweeney. Improved Outdoor Augmented Reality through Globalization. In *Doctoral Consortium, ISMAR*, 2013.

[11] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global Localization from Monocular SLAM on a Mobile Phone. *TVCG*, 20(4):531–539, 2014.

[12] D. Zou and P. Tan. CoSLAM: Collaborative Visual SLAM in Dynamic Environments. *PAMI*, 35(2):354–366, 2013.