Mobile Phone and Cloud – a Dream Team for 3D Reconstruction

Alex Locher[1] Michael Perdoch[1] Hayko Riemenschneider[1] Luc Van Gool[1,2]

[1] CVL, ETH Zurich, Switzerland [2] VISICS, KU Leuven, Belgium

Abstract

Recently, Structure-from-Motion pipelines (SfM) for the 3D reconstruction of scenes from images were pushed from desktop computers onto mobile devices, like phones or tablets. However, mobile devices offer much more than just necessary computational power. A combination of handheld device with camera, display and full connectivity entails possibilities for an on-line 3D reconstruction that would have been difficult to implement otherwise. In this work, we propose a combination of a regular mobile phone as frontend with a centralized server plus annex cloud as backend for collaborative, on-line 3D reconstruction. We illustrate few advantages of this combination of a myriad of new possibilities: First, we automatically balance computational load between the frontend and the backend depending on battery autonomy and available bandwidth. Second, we select the best of algorithms given the available resources to obtain better 3D models. Finally, we allow for collaborative modeling in order to arrive at more complete and more detailed models, especially when the objects or scenes are big. This paper presents an implementation of such a joint mobile-cloud modeling approach and demonstrates its advantages via real-life reconstructions.

1. Introduction

Over the last years, we have witnessed a steep increase in 3D developments. With the advent of 3D cinema, 3D TV, 3D printing, and cheaper or even free 3D modeling technologies, demand for 3D data production and consumption is rising. Mobile phone manufacturers are well-aware of this, and are preparing mobile phones with 3D scanning capacities. Within the computer vision community, success has pushed image-based Structure-from-Motion (SfM) pipelines (e.g. [18, 1, 32, 36]) onto mobile phones [19, 9]. Yet, the results of SfM are not always satisfactory. Like other 3D acquisition methods, SfM comes with a series of limitations. Having to make further compromises by wanting to put the entire or large part of the SfM pipeline on a mobile phone does not help. Even if the power of mobile phones will keep on increasing, there remains a plenitude of more dedicated or refined algorithms that are difficult to run onto mobile devices but could substantially improve the quality of the 3D output.

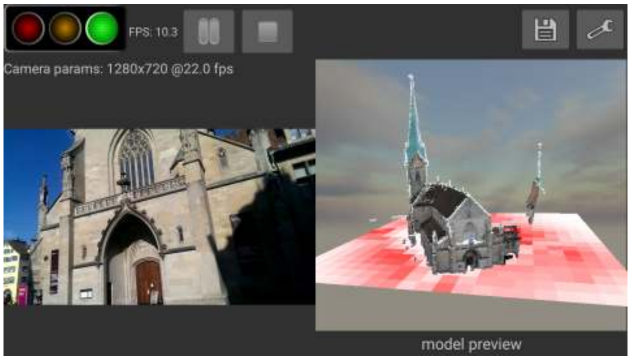


Figure 1: Screenshot of Android mobile frontend: Live camera preview (left) and current 3D reconstruction and next best view guidance (right). A traffic light symbol indicates the status of the tracker and reconstruction quality.

Hence, we propose a combination of processing on a mobile device (frontend) and processing on a server or cloud (backend) to overcome these limitations. We exploit the communication capabilities for which mobile devices have been developed in the first place. Using the backend, renders the SfM pipeline more capable than would be possible on the mobile device only. At the same time, via the use of a mobile phone as frontend, the technology can be put in the hands of millions. Rather than seeing the division of work between front and backend as fixed, it is interesting to allow for a dynamic load balancing between them, e.g. safeguarding reasonable battery autonomy and memory availability. After all, users producing 3D models may still want to make a phone call or use another app after reconstruction. Last but not least, the backend acts as the central node of a collaborative network of mobile device users. Rather than producing 3D models of the same objects repeatedly, the results produced by other users (if they agree to share) can serve as a valuable starting point. Thus, users can complete or enhance pre-existing models, or produce a model simultaneously with other users, as a group.

1.1. Contributions

In the proposed approach, we propose to combine the best of both the mobile and cloud worlds: on the one hand the immediate visual feedback and guidance during scanning and the omnipresence of modern smartphones; on the other hand, the virtually unlimited electrical power, memory, storage capacity and computational resources of the cloud server that allow to achieve better reconstruction quality and to exploit previous results. In particular, we discuss these two problems:

1. *Dynamic load balancing.* Our method automatically determines a load for the mobile client based on battery, CPU power and link bandwidth. The load is balanced according to the available resources in the mobile device and cloud, respectively.
2. *Collaborative 3D modeling of landmark buildings.* Further, we show the advantage of guiding users to take pictures that are maximally helpful in enhancing the current model, as part of the immediate visual feedback that is given. Note that such functionality can only be achieved by a system that is also aware of what others have been / are doing. It allows the system to exploit the crowd-sourcing potential of a collaborative 3D reconstruction, by giving a user an access to a prior model of the scene available in the cloud that might have been acquired by others beforehand.

1.2. Related work

Two related strands of research are closing in on each other - offline SfM and online Simultainious Localisation and Mapping (SLAM).

The offline SfM methods have progressed from cumbersome, fragile processing to the efficient handling of large-scale scenes. The initial SfM implementations were purely single desktop-based offline systems [18, 28].

Offline SfM now successfully reconstructs scenes containing hundred thousands or even millions of images [32, 8, 12]. Later, online reconstruction services decoupled the user from the powerful hardware that carried out the reconstructions [1], only needing to upload the images possibly after some simple filtering. Recently, [27, 14, 13] demonstrated an online SfM method which adds new images to existing reconstructions and builds an incremental surface model.

On the other hand, Rusinkiewicz et al. [30] demonstrated a system for real-time 3D acquisition. PTAM [16] was one of the first algorithms enabling tracking and mapping using a hand held camera in real-time for small scenes. Based on that, Kolev et al. [19] implemented the reconstruction pipeline fully on a mobile phone. SVO [7] proposed a semi-direct odometry algorithm allowing for robust tracking in real time on micro aerial vehicles. In contrast to these feature-based tracking methods, DTAM [25] uses whole image alignment, allowing a very robust tracking. The tracker is computationally expensive, but runs in real-time on a GPU. LSD-Slam [4] uses only high gradient parts of the image, which enables robust tracking of a 6 DoF camera pose in real-time on a regular laptop or even a mobile phone [31] without the need of a GPU.

Several works already incorporated a mobile client and a remote server, especially in the area of image retrieval. Girod et al. [11] worked on mobile visual search, where a current image is compared against a remote database. Image features are extracted and compressed on the phone and the server is queried for visually similar pictures. Middelberg et al. [23] localizes the 6 DoF position of a mobile phone by registering single images to a global model on a server. Forster et al. [6] used multiple Mobile Aerial Vehicles (MAVs) collaboratively for 3D scene reconstruction.

In scene reconstruction, often redundant information from similar viewpoints is available. [22, 34, 27] propose methods for selection and prediction of the most valuable additional viewpoints. ProForma [27] shows an explicit direction as visual feedback. Mauro et al. [22] use content-aware features of the 3D model and also provide a map of the best views around a 3D model.

In general, there are related works for collaborative cloud reconstructions. Hsiao et al. [15] build large reconstructions and let the users upload new images which are then integrated into these as cloud service. Middleberg et al. [24] provide a similar interface for uploading images, yet add interactive building selection from a cadastral map to simplify the retrieval. Both focus on incremental reconstruction and do not contain any prior knowledge for view guidance.

The idea of load balancing for saving power was tackled recently in [26]. To save power on a planetary rover platform they suggest to turn of a visual localization and relay on the dead reckoning odometry until the location uncertainty raises beyond requested precision. An optimal schedule for the desired precision and power savings is discussed. Contrary to these related approaches, we propose a method which balances the benefits of online and offline processing specific for each task. We propose a method for jointly solving the 3D reconstruction by task delegation and integration of prior knowledge.

2. Seamless Mobile-Cloud Reconstruction

Next, we introduce our general framework, detail its three components (frontent, session models, global model), and present three advantages of shared mobile-cloud processing (balancing of workload, delegation of tasks, exploiting high-level knowledge due to collaboration) as illustrated in Fig. 2. The framework connects the main components of the general reconstruction as follows.

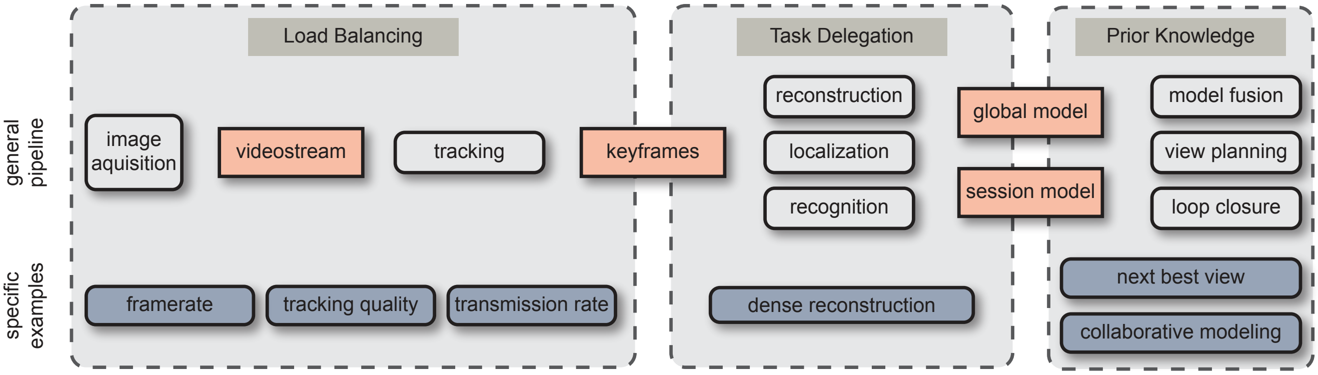


Figure 2: The proposed generic reconstruction pipeline involves several methods, which can be categorized into the 3 groups of load balancing, task delegation and prior knowledge integration. For each we give specific examples.

The **Frontend** is implemented on the mobile client and forms the interface to the user. It is responsible for acquiring an input stream of sensor data, for load balancing and for providing immediate visual feedback.

The **Session Models** contain each the reconstruction created by a single scanning session of a user. They are synchronized between the client and cloud side.

The **Global Model** stores the already reconstructed 3D content. The collaborative part integrates the individual session models with this global model, once their accurate localization is available.

A bidirectional connection between the client and the remote server allows for sophisticated load balancing. The scheme involves aspects such as the offloading of dense modeling, the delegation of object-specific reconstruction to dedicated algorithms (task delegation), and the exploitation of the full 3D environment for visual guidance (prior knowledge). We propose three novelties in the context of such mobile-cloud framework, namely load balancing, task delegation and prior knowledge integration.

The **Load Balancing** divides the work depending on the available resources (battery, CPU power, data link bandwidth).

A **Task Delegation** scheme acts as a higher level load balancing layer and tries to assign reconstruction tasks to the platform and algorithm best suited.

Prior Knowledge brings additional benefits to the user. Having the cloud in the background, the user gets access to a collaborative environment where sharing can simplify her task substantially. The system points out how the user can best contribute.

We demonstrate the capabilities of such framework with an implementation that we test on several examples.

3. The 3D Dream Team in Action

The fully working implementation consists of a mobile smartphone application and server counterpart, incorporating several state-of-the-art components. We introduce the most important parts here and then give more details in following sections.

3.1. Mobile Client

An Android application on a smartphone or tablet acts as the mobile client, incorporating the mobile frontend and additional layers for communication and load balancing. Its user interface (Fig. 1) and parts of the visualisation are written in Java, whereas the vision algorithms are implemented in native C++ code and all computations are currently performed on the phone's CPU. The tracker of the publicly available LSD SLAM [4] continuously tracks the 6 DoF camera position, with respect to selected keyframes, using semi-dense image alignment. While tracking, the depth of high gradient pixels in the active keyframe are refined by many small baseline comparisons. As soon as the distance between the camera and the closest keyframe is large enough, a new keyframe is initialized by projecting depth values of the last keyframe into the current frame.
Keyframes with depth maps, which are not refined any longer, are incorporated into a pose graph, forming the client's session model. Individual nodes of the pose graph are connected by an estimated 7 DoF similarity transform and optimized using general graph optimization. The communication module exchanges keyframes, localization and visualisation data, depending on the current load balancing scheme, with the remote server.

3.2. Cloud Server

In the cloud, the reconstruction server listens for incoming connections of mobile clients. After successful identification, a session id is generated and a new session model is started. Keyframes and pose constraints from the client are saved and the corresponding pose graph is created. Additional constraints between the new and existing data are searched by local features matching. Matching candidates are evaluated by incorporating the pose priors and image retrieval applied to the existing keyframes in the session model. The server application is also implemented as a C++ application and links to the g2o [20] optimisation. Feature extraction and matching are performed using a GPU implementation [35]. Reconstruction and registration are realized using VisualSFM [37] controlled over a socket.

4. Load balancing

The load balancing layer enables to change parameters of the frontend on the fly. Its goal is to optimize the user experience given the computation resources available, to minimize power consumption, and to achieve the fastest possible model registration in the cloud. The load balancing is based on low level parameters (battery b, connectivity c and cpu power p) and controls the keyframe transmission and tracking schemes. The following input parameters are used: the battery level is given in percents of charge left, $b \in [0, 1]$, the connectivity is measured using the current connection standard $c \in \{E, 3g, 4g, \mathit{WLAN}\}$ and the cpu power is modeled by the number of cores ($p \in \mathbb{N}$).

4.1. Tracking scheme

Controlling the tracking allows to balance between power consumption and usability. We define three different tracking strategies: *NONE*, *SLOW* and *FAST*. The strategy is chosen by the load balancing as:

$$Q_T(b,p) = \begin{cases} \mathit{NONE}, & \text{if} \quad p = 1 \\ \mathit{SLOW}, & \text{if} \quad p > 1 \wedge b < 0.15 \\ \mathit{FAST}, & \text{otherwise} \end{cases} \qquad (1)$$

In normal operation mode ($Q_T = \mathit{FAST}$) the tracker runs at a resolution of 320 × 240 pixels and at the maximal achievable framerate. In order to respect the mobile device lifetime, tracking quality is reduced on low battery levels ($Q_T = \mathit{SLOW}$). This is achieved by reducing the input image size to 240 × 160 pixels and limiting the framerate to 8 fps. In case of a low end phone (with a single CPU), tracking is disabled completely and the user is prompted to take individual photographs of the scene ($Q_T = \mathit{NONE}$), making sure that there is enough overlap. In this case, all of the computations are delegated to the server backend. This saves a lot of computational power but also potentially reduces the reconstruction quality.
In order to find the optimal image sizes, we measured the average framerate of the tracker on a test sequence using our system on different devices (see Table 2). The smallest frame size offers the highest framerate, but due to the low image quality, tracking quality is bad. Similar to the authors of [31] we found a frame size of 320 × 240 to be the best compromise for the mobile phone. Table 1 shows the per - scheme average power consumption of a sample sequence.

Table 1: Average power consumption of the different tracking schemes measured using a HTC One mobile phone.

tracking scheme	power consumption	relative
NONE	2.5 mW	100 %
SLOW	1.7 mW	68 %
FAST	0.7 mW	28 %

4.2. Transmission scheme

The rate and resolution at which keyframes are transferred to the server influence the reconstruction quality of the server-side session model as well as the chance of a successful registration to the global model. In reality, the communication channel has limited bandwidth and constant transmission also drains the battery of the mobile phone. By controlling the keyframes' image resolution, we can adjust the transmission time to the current network status. The active transmission scheme Q_C is a function of the connectivity c and battery level b:

$$Q_C(c,b) = \begin{cases} \mathit{LOW}, & \text{if} \quad b < 0.15 \vee c \in \{E\} \\ \mathit{MEDIUM}, & \text{if} \quad b \geq 0.15 \wedge c \in \{3g\} \\ \mathit{HIGH}, & \text{otherwise} \end{cases} \qquad (2)$$

The low connectivity strategy ($Q_C = \mathit{LOW}$) only transmits very small thumbnails of images to the server, which are registered to the global model but not used for model extension. The high resolution image is stored on the internal SD card and eventually transferred to the server as the connection quality increases. In the medium connectivity scenario ($Q_C = \mathit{MEDIUM}$), keyframes with resolution of 640×360 are transferred to the server. The increased image resolution allows a more accurate registration and an additional model extension on the cloud. The higher resolution images are again stored on the device's SD card. In situations with good connections ($Q_C = \mathit{HIGH}$), a high resolution image (1280×720 pixels) is immediately transferred to the server, where it can contribute to the extension of the global model. Table 3 summarizes typical transmission times for images in the corresponding scheme.

5. Task delegation

In addition to the low level load balancing, acting mainly on the mobile frontend, we also introduce a high level task scheduling on the algorithmic layer. Its task is to assign the

Table 2: Average tracking framerate of our system, measured for different devices and frame sizes. The dimension highlighted in green offers the best compromise in terms of tracking quality and power consumption and are therefore selected by the load balancing. The last column shows the chosen camera default framerate, being slightly higher than the tracker's rate.

Device	CPU	Frame Size				Camera FPS
		240 × 160	320 × 240	640 × 480	1280 × 720	
HTC One	4 × 1.7 GHz	24.5	17.2	6.4	2.1	20.0
Galaxy Note 8.0	4 × 1.6 GHz	22.7	15.8	3.6	1.6	17.0
Google Nexus 9	2 × 2.3 GHz	24.2	21.2	4.7	2.1	22.0
Laptop Intel i7-3520M	4 × 2.9 GHz	72.8	56.4	19.2	7.1	—

Table 3: Transmission time for different images and link speeds (green highlighted are active in the system).

Image Size	JPG Size	Link			
		E	3g	4g	WiFi
320 × 180	28 KB	1.9s	0.6s	60ms	4ms
640 × 360	86 KB	5.7s	1.7s	170ms	12ms
1280 × 720	257 KB	17.1s	5.1s	510ms	38ms

reconstruction of scene parts to the algorithm best suited for those parts and to the platform providing the necessary resources. The concept holds for any kind of specific reconstruction algorithms, but is demonstrated on the resource demanding dense reconstruction.

Load balancing typically tries to offload heavy tasks to the backend. In our system only a coarse tracking combined with keyframe selection is performed on the mobile phone. Although dense modeling has been shown possible on the phone, the system is bound to offload this heavy task to the cloud, respecting battery lifetime and improving results. For visualisation, the sparse point cloud of the session model is fed into the CMVS & PMVS2 [10] pipeline delivering high quality 3D pointclouds.

Experiment
We illustrate the capabilities of the dense modeling in Fig. 3. The first visual feedback from the LSD SLAM's pose graph already gives a good impression of the scene but is very noisy (Fig. 3a). Thanks to the basically unlimited resources on the server side, the feature detection and matching can be massively parallelized and the sparse point cloud can be reconstructed almost on the fly. The textured surface (Fig. 3b) based on the sparse point cloud already provides much more detailed feedback, is less noisy and can be calculated in seconds on the server. Finally, the dense model (Fig. 3c) of a large scene is computed on the server and pushed to the client. Due to the offloading, the densification step is not limited in size and saves power by not running on the client's CPU.

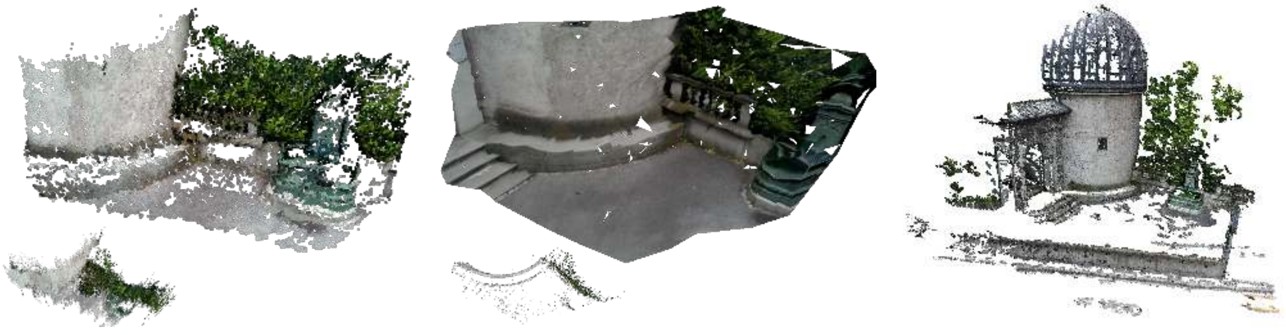
6. Prior Knowledge Integration

Another corner stone in our mobile-cloud framework is the use of prior knowledge that is available when previous users have made their results available already. By a registration of the session model to the global model, the cloud provides reconstruction results of parts not modeled by the user herself and, furthermore, extends the global 3D model with the new session information coming from that user. Moreover, a view planning algorithm [22] creates a coherent map of the environment and guides the user towards the locations from where best to take additional images in order to best complement the current state of the model.

6.1. Localization and Integration of Session Models

So far, the server-side session model was composed only from the local observations made by a single the mobile client. The immense size of cloud storage allows to build and store images and 3D models of substantially larger scaled scenes, such as city-scale models, detailed models of landmarks, statue collections, etc. To use this offline storage, first a registration or localization of the current session model must be performed. With a substantial part of the SfM pipeline already running on the mobile phone, typically the load balancing decides that a dedicated cloud-based image retrieval algorithm is to be used. Our implementation employs a bag-of-words specific object retrieval system with spatial verification. To initialize the system, a visual vocabulary was trained on a representative sample of images from a city-scale street-level dataset.

In the localization phase, SIFT features from the keyframe images computed for the session model SfM are labeled using an approximate nearest neighbor to the closest visual words. A fast TF-IDF scoring is applied to compute the similarity with all images in the database and a shortlist of the most similar ones is geometrically verified for a consistent affine transformation between the query image and each of the database images. Finally, the images are reranked using the number of consistent matches as a scoring



(a) LSD depth frames (top view in small)

(b) textured surface (top view in small)

(c) dense cloud

Figure 3: The pose graph with asscociated depth images delivers a fast but noisy feedback on the phone. Although theoretically possible on the phone, the demanding dense reconstruction is offloaded to the server. A textured surface based on the sparse point cloud already yields good visual feedback whereas the dense point cloud gets available with a small latency.

function. The relevant part is then selected by considering the top candidate camera positions in the SfM model as well as their 2D-3D matches.

At this stage the session model from the mobile user is coarsely localized in the same reference frame as the global model on the cloud server. Hence, it is known how to use the session model to extend the 3D environment. For keyframe-wise fine registration to the global model, a 3D submodel around the coarse location of the keyframe is extracted from the global 3D model and loaded into VisualSFM. The camera poses of the submodel are kept fixed, while the server side session model is merged into it and optimized with an incremental bundle adjustment [36]. For the fusion of global and session model, the optimization is performed over all cameras in the local submodel. Note that due to the extraction of a submodel from the global model, the fusion algorithm does not dependent on the overall size of the global model. Finally, the new globally registered keyframes and poses can be integrated into the global model and made available to other users.

Experiments

The reconstruction and registration speed are demonstrated on a real-world scenario in Fig. 4. The sequence was recorded with a mobile phone and a global server model was created using a Canon EOS 70D. For the experiment, the video was replayed on the mobile phone connected to the remote server over WiFi. Timings are average measurements for this particular sequence. We distinguish between two cases, one where we are able to register the local model and one where a registration is not possible. The latter case corresponds to a scene not covered by any global model.

Right after starting the scan session, a still noisy point cloud from the SLAM gives a first impression of the scene (A). After three seconds, the first two keyframes are transmitted to the server. Using the retrieval engine, close images in the global model are identified and matched against the new images and matches are used to localize them. The whole localization step, including indexing, retrieval, submodel extraction and additional bundle adjustment takes nine seconds on the server. A compressed version of the global model can be presented to the user, only 18 seconds after session start (B). While the user is still scanning, the computation of a dense model is started on the server. The densification step takes 55 seconds for 17 new keyframes. The final refined global model becomes available after 78 seconds (E). In the case of unsuccessful global registration, the server still reconstructs the local scene. 23 seconds after the session start, a more accurate sparse cloud extends the information of the SLAM system (C). A server side densification provides a dense point cloud of only the local scene after 61 seconds (D).

6.2. Visual Feedback and View Guidance

An important part of the user experience is the quality and latency of the visual feedback. We provide multiple types of visualization depending on the cloud's availability and registration status. A successful registration to the global model makes it possible to display model parts never scanned by the user himself. In addition, the user is visually guided towards poorly covered scene parts in order to improve them. For this instance, a dedicated next best view algorithm [22] runs on the server after every significant update. A colored grid is transferred to the client and displayed simultaneously with the 3D model. It indicates where to best take additional images, in order to extend the existing 3D model. Please note that next best view guidance can also be run entirely on the phone, yet only when the session model is finely registered to the global model the true collaborative power of our mobile-cloud reconstruction comes to bear.

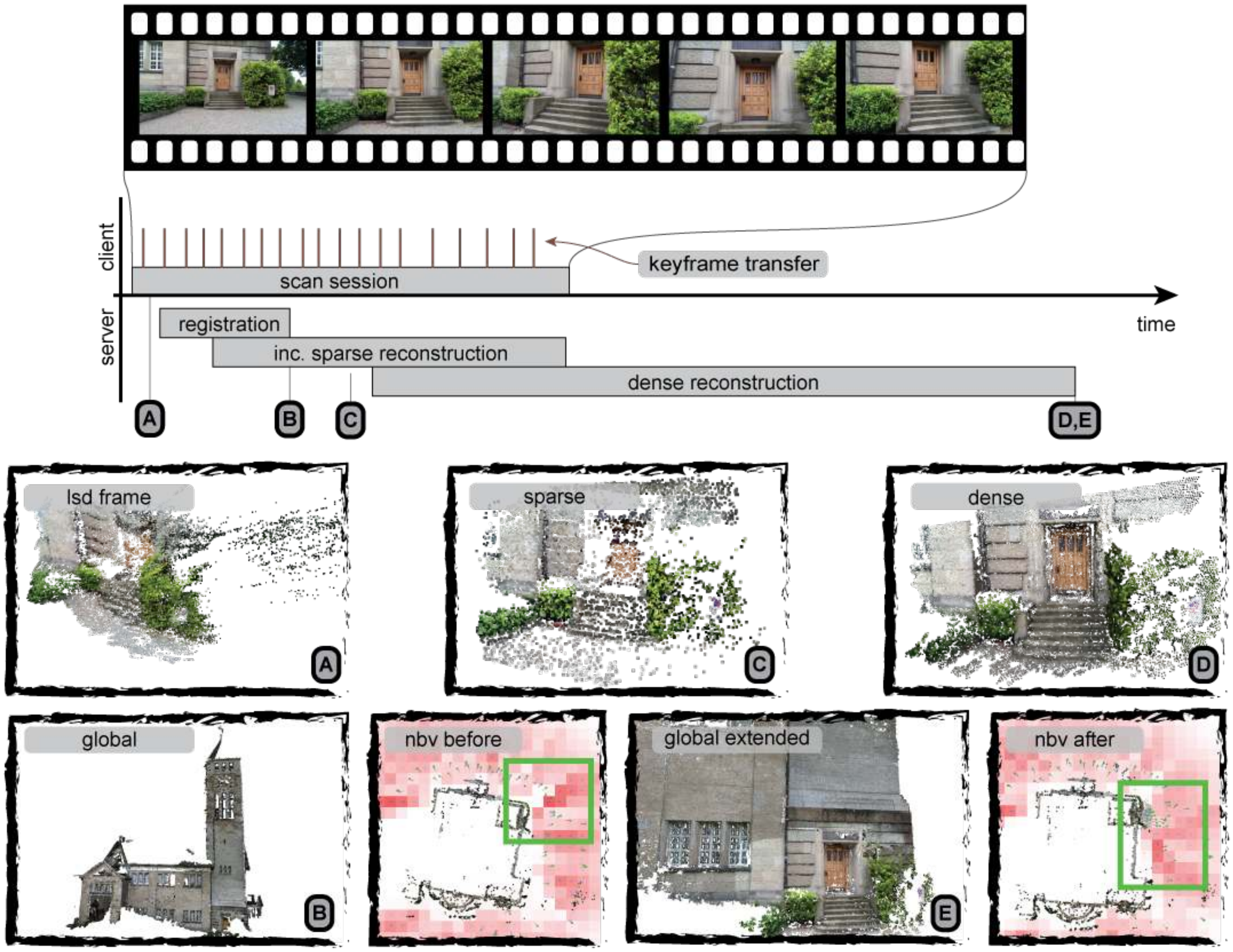


Figure 4: A timeline of a sample sequence processed by our system. During the scan session on the mobile phone, keyframes (indicated by peaks) are transferred to the remote server, where an immediate registration to a global model is started. In parallel a sparse reconstruction only of the session model delivers an accurate sparse point cloud, which again is densified on the cloud. The middle row shows the visual feedback, available to the client at different time steps. If global registration succeeds, a dense version of the whole surrounding is presented immediately and afterwards extended by the new images (bottom row). A next best view (nbv) map, indicates good spots for additional images with red peaks.

Experiment

The next best view map for the sequence in Fig. 4 illustrates its behaviour. Before the global model is extended, the top left corner shows red peaks corresponding to good spots from where to take additional views. After merging views from the session model the corresponding spots vanish and new peaks arise closeby. The new proposed spot matches the incomplete facade of the church, visible in Fig. 4 B. The calculation of the next best view on the server took 166 seconds. For comparison, we run the same computation on the phone itself, being 20 times slower on average.

7. Conclusion

We presented a 3D reconstruction framework seamlessly tying together a mobile client application and a powerful cloud solution enabling collaborative reconstruction of a potentially large scene using everyday smartphones. The combination of multi-layered, fast reconstruction feedback with next best view propositions helps the user to optimize the reconstruction result. A separation between session and global models allows to handle multiple users in parallel and to extend existing reconstructions. A load balancing scheme ensures that valuable resources such as battery power are used wisely. A high level task delegation assigns reconstruction tasks to methods and platforms most appropriate for the job. We demonstrated the capabilities of the approach through an implementation that combines a real-time Android phone part with a server counterpart. This is a proof-of-concept paper and there is ample room for extensions, e.g. efficient surface reconstruction [2, 3], scene to speech transcription [33], object specific reconstruction [5], or semantic segmentation [21, 29] or decomposition [17].

Acknowledgment. The European Research Council (ERC) project VarCity (No.273940) and H2020 project REPLICATE (No.687757) supported this work.

References

[1] ARC 3D Webservice. http://www.arc3d.be/.

[2] A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool. Fast, Approximate Piecewise-Planar Modeling Based on Sparse Structure-from-Motion+Superpixels. In *CVPR*, 2014.

[3] A. Bodis-Szomoru, H. Riemenschneider, and L. Van Gool. Superpixel Meshes for Fast Edge-Preserving Surface Reconstruction. In *CVPR*, 2015.

[4] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.

[5] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. 32(9):1627–1645, 2010.

[6] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza. Collaborative monocular slam with multiple micro aerial vehicles. In *IROS*, 2013.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *ICRA*, 2014.

[8] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and Marc-Pollefeys. Building rome on a cloudless day. In *ECCV*. 2010.

[9] D. Fritsch and M. Syll. Photogrammetric 3d reconstruction using mobile imaging. In *IS&T SPIE Electronic Imaging Conference*, 2015.

[10] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. 32(8):1362–1376, 2010.

[11] B. Girod, V. Chandrasekhar, D. M. Chen, N.-M. Cheung, R. Grzeszczuk, Y. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham. Mobile visual search. *Signal Processing Magazine, IEEE*, 28(4):61–76, 2011.

[12] J. Heinly, J. L. Schönberger, E. Dunn, and J.-M. Frahm. Reconstructing the world in six days. In *CVPR*, 2015.

[13] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental surface extraction from sparse structure-from-motion point clouds. *BMVC*, 2013.

[14] 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*, 2012.

[15] D. Hsiao, N. Tabing, Z. Popovic, K. Tuite, and N. Snavely. Photocity: Training experts at large-scale image acquisition through a competitive game. In *CHI*, 2011.

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

[17] N. Kobyshev, A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool. Architectural Decomposition for 3D Landmark Building Understanding. 2016.

[18] R. Koch, M. Pollefeys, and L. Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *ECCV*, 1998.

[19] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning mobile phones into 3d scanners. In *CVPR*, 2014.

[20] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and Burgard. G2o: A general framework for graph optimization. *ICRA*, 2011.

[21] A. Martinović, J. Knopp, H. Riemenschneider, and L. Van Gool. 3D All The Way: Semantic Segmentation of Urban Scenes from Start to End in 3D. In *CVPR*, 2015.

[22] M. Mauro, H. Riemenschneider, A. Signoroni, R. Leonardi, and L. Van Gool. A unified framework for content-aware view selection and planning through view importance. In *BMVC*, 2014.

[23] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-dof localization on mobile devices. In *ECCV*, 2014.

[24] S. Middleberg, O. Untzelmann, T. Sattler, and L. Kobbelt. A scalable collaborative online system for city reconstruction. In *ICCVWS*, 2013.

[25] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, 2011.

[26] P. Ondruska, C. Gurau, L. Marchegiani, C. H. Tong, and I. Posner. Scheduled perception for energy-efficient path following. In *ICRA*, 2015.

[27] Q. Pan, G. Reitmayr, and T. Drummond. Proforma: Probabilistic feature-based on-line rapid model acquisition. In *BMVC*, 2009.

[28] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Hand-held acquisition of 3d models with a video camera. In *3DIM*, 1999.

[29] H. Riemenschneider, A. Bodis-Szomoru, J. Weissenberg, and L. Van Gool. Learning Where To Classify In Multi-View Semantic Segmentation. In *ECCV*, 2014.

[30] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *Siggraph*, 2002.

[31] T. Schöps, J. Engel, and D. Cremers. Semi-dense visual odometry for AR on a smartphone. In *ISMAR*, 2014.

[32] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *Siggraph*, 2006.

[33] J. Weissenberg, M. Gygli, H. Riemenschneider, and L. Van Gool. Navigation using Special Buildings as Signposts. In *MapInteract*, 2014.

[34] S. Wenhardt, B. Deutsch, E. Angelopoulou, and H. Niemann. Active visual object reconstruction using d-, e-, and t-optimal next best views. In *CVPR*, 2007.

[35] C. Wu. Siftgpu: A gpu implementation of scale invariant feature transform (sift), 2007.

[36] C. Wu. Towards linear-time incremental structure from motion. In *3DV*, 2013.

[37] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *CVPR*, 2011.