Live Metric 3D Reconstruction on Mobile Phones

Petri Tanskanen, Kalin Kolev, Lorenz Meier, Federico Camposeco, Olivier Saurer, Marc Pollefeys

ETH Zurich, Switzerland

Abstract

In this paper, we propose a complete on-device 3D reconstruction pipeline for mobile monocular hand-held devices, which generates dense 3D models with absolute scale on-site while simultaneously supplying the user with real-time interactive feedback. The method fills a gap in current cloud-based mobile reconstruction services as it ensures at capture time that the acquired image set fulfills desired quality and completeness criteria. In contrast to existing systems, the developed framework offers multiple innovative solutions. In particular, we investigate the usability of the available on-device inertial sensors to make the tracking and mapping process more resilient to rapid motions and to estimate the metric scale of the captured scene. Moreover, we propose an efficient and accurate scheme for dense stereo matching which allows to reduce the processing time to interactive speed. We demonstrate the performance of the reconstruction pipeline on multiple challenging indoor and outdoor scenes of different size and depth variability.

1. Introduction

The flexible and accurate generation of 3D models of real-world environments has been a long-term goal in computer vision. Research efforts on 3D content creation from still images has reached a certain level of maturity and has emerged to popular industrial solutions like Autodesk's 123D Catch. While high-quality 3D models can be obtained with such systems, the selection of an image set, which ensures the desired accuracy and completeness, is not a trivial task. Occlusions, complex reflectance properties and shading effects often lead to failure in the reconstruction process but their appearance is difficult to predict in advance, especially for non-experts. This challenge is addressed by monocular real-time capable systems which can provide useful feedback to the user in the course of the reconstruction process and guide his movements. Impressive results were obtained with video cameras [7] and depth sensors [15, 8]. However, those systems require massive processing resources like multi-core CPUs and powerful GPUs. As a result, their usability is limited to desktop computers and high-end laptops, which precludes applications of casual capture of 3D models in the wild. Moreover, the produced 3D models are determined only up to an overall scale and are not provided in metric coordinates. This burdens their applicability in areas where precise physical measurements are needed.

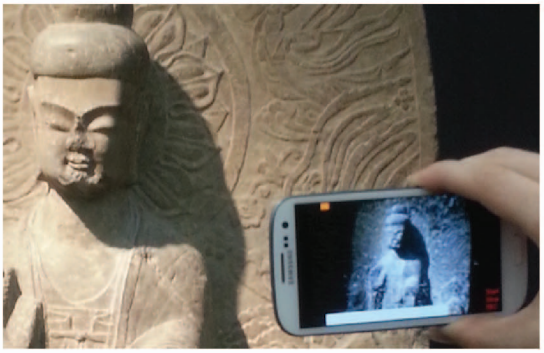

Figure 1. Live 3D reconstruction in a museum. The smartphone display shows the dense 3D model registered to the object. Full results in Fig. 7.

In the last few years, remarkable progress was made with mobile consumer devices. Modern smartphones and tablet computers offer multi-core processors and graphics processing cores which open up new application possibilities. Additionally, they are equipped with micro-electrical sensors, capable of measuring angular velocity and linear acceleration. The design of methods for live 3D reconstruction able to make use of those developments seems a natural step. While first attempts for interactive 3D reconstruction on smartphones have already been presented [9, 13], their applicability is limited and their performance is still far from that of desktop systems.

In this paper, we propose the first dense stereo-based system for live interactive 3D reconstruction on mobile phones (see Fig. 1). We leverage the resources offered by current smartphones and address the challenges posed by the underlying hardware limitations with multiple novel solutions. In particular:

- The system is fully automatic and does not require markers or any other specific settings for initialization.

- We perform feature-based tracking and mapping in real time but leverage full inertial sensing in position and orientation to estimate the metric scale of the reconstructed 3D models and to make the process more resilient to sudden motions.

- The system offers an interactive interface for casual capture of scaled 3D models of real-world objects by non-experts. The approach leverages the inertial sensors to automatically select suitable keyframes when the phone is held still and uses the intermediate motion to calculate scale. Visual and auditory feedback is provided to enable intuitive and fool-proof operation.

- We propose an efficient and accurate multi-resolution scheme for dense stereo matching which makes use of the capabilities of the GPU and allows to reduce the computational time for each processed image to about 2-3 seconds.

2. Related Work

Our work is related to several fields in computer vision: visual inertial fusion, simultaneous localization and mapping (SLAM) and image-based modeling.

Visual inertial fusion is a well established technique [1]. Lobo and Dias align depth maps of a stereo head using gravity as vertical reference in [6]. As their head is calibrated, they do not utilize linear acceleration to recover scale. Weiss *et al*. [23] developed a method to estimate the scaling factor between the inertial sensors (gyroscope and accelerometer) and monocular SLAM approach and the offsets between the IMU and the camera. Porzi *et al*. [12] demonstrated a stripped-down version of a camera pose tracking system on an Android phone where the inertial sensors are utilized only to obtain a gravity reference and frame-to-frame rotations.

Recently Li and Mourikis demonstrated impressive results on visual-inertial visual odometry, without reconstructing the environment [5].

Klein and Murray [3] proposed a system for real-time parallel tracking and mapping (PTAM) which was demonstrated to work well also on smartphones [4]. Thereby, the maintained 3D map is built from sparse point correspondences only. Newcombe *et al*. [7] perform tracking, mapping and dense reconstruction on a high-end GPU in real time on a commodity computer to create a dense model of a desktop setup. Their approach makes use of general purpose graphics processing but the required computational resources and the associated power consumption make it unsuitable for our domain.

As the proposed reconstruction pipeline is based on stereo to infer geometric structure, it is related to a myriad of works on binocular and multi-view stereo. We refer to the benchmarks in [16], [17] and [19] for a representative list. However, most of those methods are not applicable to our particular scenario as they don't meet the underlying efficiency requirements. In the following, we will focus only on approaches which are conceptually closely related to ours.

Building upon previous work on reconstruction with a hand-held camera [10], Pollefeys *et al*. [11] presented a complete pipeline for real-time video-based 3D acquisition. The system was developed with focus on capturing large-scale urban scenes by means of multiple video cameras mounted on a vehicle. A method for real-time interactive 3D reconstruction was proposed by Stuehmer *et al*. [20]. Thereby, a 3D representation of the scene is obtained by estimating depth maps from multiple views and converting them to triangle meshes based on the respective connectivity. Another approach for live video-based 3D reconstruction was proposed by Vogiatzis and Hernandez [22]. Here, the captured scene is represented by a point cloud where each generated 3D point is obtained as a probabilistic depth estimate by fusing measurements from different views. Even though the aforementioned techniques cover our context, they are designed for high-end computers and are not functional on mobile devices due to some time-consuming optimization operations.

Recently, the first works on live 3D reconstruction on mobile devices appeared. Wendel *et al*. [24] rely on a distributed framework with a variant of [4] on a micro air vehicle. All demanding computations are performed on a separate server machine that provides visual feedback to a tablet computer. Pan *et al*. [9] demonstrated an interactive system for 3D reconstruction capable of operating entirely on a mobile phone. However, the generated 3D models are not very precise due to the sparse nature of the approach. Prisacariu *et al*. [13] presented a shape-from-silhouette framework running in real time on a mobile phone. Despite the impressive performance, the method suffers from the known weaknesses of silhouette-based techniques, e. g. the inability to capture concavities. In contrast, the proposed system does not exhibit these limitations since it relies on dense stereo.

3. System Overview

Our system consists of three main blocks: *inertial tracking*, *visual pose estimation* and *dense 3D modeling*, as depicted in Fig. 2. All three blocks operate asynchronous and thus allow us to optimally make use of the multi-core capabilities of the device. We take two main input streams: camera frames with resolution of 640×480 at tracking rates typically between 15-30 Hz and inertial sensor infor-

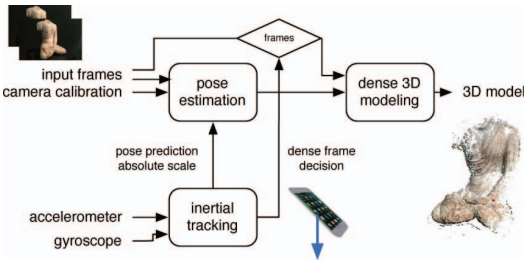

Figure 2. Interconnections between the main building blocks.

mation (angular velocity and linear acceleration) at 200 and 100 Hz respectively. The inertial tracker provides camera poses which are subsequently refined by the visual tracking module. The dense 3D modeling module is supplied with images and corresponding full calibration information at selected keyframes from the visual tracker as well as metric information about the captured scene from the inertial tracker. Its processing time is typically about 2-3 seconds per keyframe. The system is triggered automatically when the inertial estimator detects a salient motion with a minimal baseline. The final output is a 3D model in metric coordinates in form of a colored point cloud. All components of the system are explained in more detail in the following sections.

4. Visual Inertial Scale Estimation

Current smartphones are equipped with a 3D gyroscope and accelerometer, which produce (in contrast to larger inertial measurement units) substantial time-dependent and device-specific offsets, as well as significant noise. To estimate scale, we first need to estimate the current world to body/camera frame rotation R_B and the current earth-fixed velocity and position using the inertial sensors. The estimation of this rotation is achieved through a standard Extended Kalman Filter. As the magnetometer and GPS are subject to large disturbances or even unavailable indoors as well as in many urban environments, we rely solely on the gyroscope and update the yaw angle with visual measurements m_B. We scale the gravity vector g_B to the unit-length vector z_B and estimate y_B and x_B using the additional heading information

$$r_{zB} = \frac{g_B}{\|g_B\|}, \quad r_{yB} = \frac{r_{zB} \times m_B}{\|r_{zB} \times m_B\|}, \quad r_{xB} = r_{yB} \times r_{zB}, \tag{1}$$

with R_B and dynamics given as

$$R_B = [r_{xB}, r_{yB}, r_{zB}] \in \mathcal{SO}(3), \quad \dot{R}_B = \vec{\omega} R. \tag{2}$$

The filter prediction and update equations are given as

$$\hat{R}_B = e^{\vec{\omega} dt} \bar{R}_B^-, \tag{3}$$

$$\bar{r}_{iB}^+ = \hat{r}_{iB} + L_{ik}(z_i - \hat{r}_{iB}) \text{ with } i \in (x, y, z), \tag{4}$$

where the Kalman gain matrix L_k is computed in every time step with the linearized system.

The camera and IMU are considered to be at the same location and with the same orientation. In the case of orientation, this is valid since both devices share the same PCB. As for the case of the location, this is a compromise between accuracy and simplicity. For the proposed framework, neglecting the displacement between sensors did not noticeably affect the results.

We initialize the required scale for visual-inertial fusion by first independently estimating motion segments. In order to deal with the noise and time-dependent bias from the accelerometer, an event-based outlier-rejection scheme is proposed. Whenever the accelerometer reports significant motion, we create a new displacement hypothesis $\vec{x}$. This is immediately verified by checking a start and stop event in the motion. These are determined given that for sufficiently exciting handheld motion, the acceleration signal will exhibit two peaks of opposite sign and significant magnitude. A displacement is then estimated and compared to the displacement estimated by vision ($\vec{y}$) at the start and stop events, yielding a candidate scale. Due to visual or inertial estimation failures, outlier rejection is needed. Each new measurement pair is stored and the complete set is re-evaluated using the latest scale by considering a pair as inlier if $\|\vec{x}_i - \lambda \vec{y}_i\|$ is below a threshold. If the new inlier set is bigger than the previous one, a new scale λ is computed in the least-squares sense using the new set I as

$$\arg\min_{\lambda} = \sum_{i \in I} \|\vec{x}_i - \lambda \vec{y}_i\|^2. \tag{5}$$

Otherwise, the displacements are saved for future scale candidates.

As soon as the scale estimation converges, we can update the inertial position with visual measurements. In addition to providing an estimate of the scene scale, we produce a filtered position estimation as show in Fig 3. This can be leveraged to process frames at lower rates or to mitigate intermediate visual tracking issues e.g. due to motion blur. Since the sample rate of the accelerometer is higher than the frame rate of the camera, we predict the position of the phone with each new accelerometer sample and update with the visual information whenever a new measurement is available. With every new IMU sample, the accelerometer data is rotated and the gravity is accounted for in the inertial frame. This acceleration is integrated using Velocity Verlet, which is in turn used for a decaying velocity model

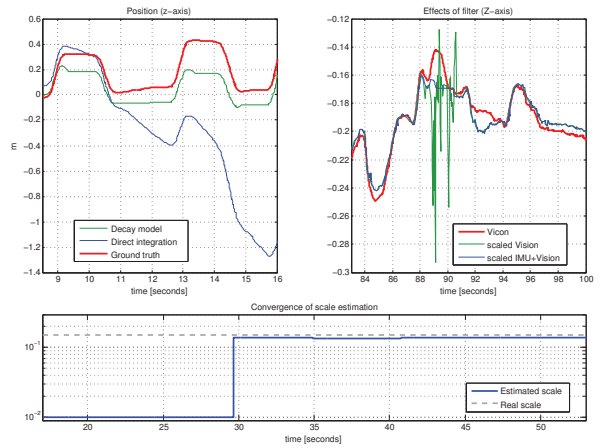

Figure 3. Left: simple inertial prediction and decaying velocity vs ground truth. Right: visual-inertial estimate allows to partially reject tracking losses. Bottom: Convergence of scale estimation.

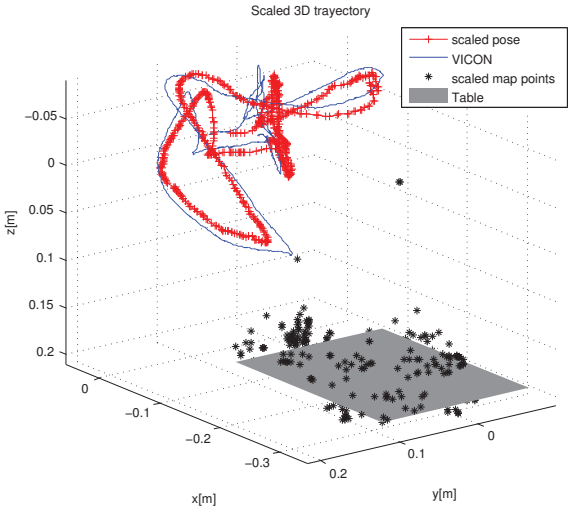

Figure 4. Visual inertial pose estimate vs. ground truth.

of handheld motion

$$\vec{v}_I^{\,k+1} = \vec{v}_I^{\,k} + \tau \Delta t R_B \left(\vec{a}_B^{\,k} - g_B \right). \tag{6}$$

Here τ accounts for timing and sensor inaccuracies (inherent of the operating system available on mobile phones) by providing a decaying velocity model, preventing unwanted drift at small accelerations (see Fig 3). To adapt to the visual data, it is first scaled to metric units using λ and then fused with the inertial prediction using a simple linear combination based on the variances of both estimations

$$\vec{x_f} = \kappa \left(\sigma_v^{-2} \lambda \vec{x_v} + \sigma_i^{-2} \vec{x_i} \right). \tag{7}$$

Here the subscripts f, v and i denote fused, vision and inertial position estimates, respectively, and κ is the normalizing factor.

The visual updates become available with a time offset, so we need to re-propagate the predicted states from the point, at which the vision measurement occurred, to the current one [23]. This is done by storing the sates in a buffer and, whenever a visual measurement arrives, looking back for the closest time-stamp in that buffer, updating and propagating forward to the current time.

Fig 4 shows the results of the combined vision and inertial fusion in a freehand 3D motion while tracking a tabletop scenario. It is evident that scale and absolute position are correctly estimated throughout the trajectory.

To evaluate the estimated scale accuracy, metric reconstruction of a textured cylinder with a known diameter was performed. In a qualitative evaluation with multiple tries, the scale was estimated to have an error of up to 10-15%. This is mostly due to the inaccuracy in the magnitude of the measurements of the consumer-grade accelerometer on the device. It should be noted that the accuracy of those measurements could be improved by calibrating the accelerometer with respect to the camera beforehand, but such investigations are left for future work.

5. Visual Tracking and Mapping

5.1. Two View Initialization

The map is initialized from two keyframes. ORB features [14] are extracted from both frames and matched. Outliers are filtered out by using the 5-point algorithm in combination with RANSAC. After that, relative pose optimization is performed and the point matches are triangulated. The rest of the initialization follows the design of [3]: In order to get a denser initial map, FAST corners are then extracted on four resolution levels and for every corner a 8x8 pixel patch at the respective level is stored as descriptor. The matching is done by comparing the zero-mean sum of squared differences (ZSSD) value between the pixel patches of the respective FAST corners along the epipolar line. To speed up the process, only the segment of the epipolar line is searched that matches the estimated scene depth from the already triangulated points. After the best match is found, the points are triangulated and included to the map which is subsequently refined with bundle adjustment. Since the gravity vector is known from the inertial estimator, the map is also rotated such that it matches the earth inertial frame.

5.2. Patch Tracking and Pose Refinement

The tracker is used to refine the pose estimate from the inertial pose estimator and to correct drift. For every new camera frame FAST corners are extracted and matched with

the projected map points onto the current view using the inertial pose. The matching is done by warping the 8x8 pixel patch of the map point onto the view of the current frame and computing the ZSSD score. It is assumed that its normal is oriented towards the camera that observed it for the first time. For computing the warp the appropriate pyramid level in the current view is selected. The best matching patch in a certain pixel radius is accepted. The matches are then optimized with a robust Levenberg-Marquart absolute pose estimator giving the new vision-based pose for the current frame. If for some reason the tracking is lost the small blurry image relocalization module from [3] is used.

5.3. Sparse Mapping

New keyframes are added to the map if the user has moved the camera a certain amount or if the inertial position estimator detects that the phone is held still after salient motion. In either case, the keyframe is provided to the mapping thread that accepts the observations of the map points from the tracker and searches for new ones. To this end, a list of candidates is created from non maximum suppressed FAST corners that have a Shi-Tomasi score [18] above a certain threshold. To minimize the possibility that new points are created at positions where such already exist, a mask is created to indicate the already covered regions. No candidate is added to the map if its projection is inside a certain pixel radius. Since the typical scene consists of an object in the middle of the scene, only map points that were observed from an angle of 60 degrees or less relative to the current frame are added to this mask. This allows to capture both sides of the object but still reduces the amount of duplicates.

Similar to [3], the mapper performs bundle adjustment optimization in the background. Its implementation is based on the method using the Schur complement trick that is described in [2]. After a keyframe is added, a local bundle adjustment step with the closest 4 keyframes is performed. With a reduced priority, the mapper optimizes the keyframes that are prepared for the dense modeling module. Frames, that have already been provided to the module, are marked as fixed. With lowest priority, the mapping thread starts global bundle adjustment optimization based on all frames and map points. This process is interrupted if new keyframes arrive.

6. Dense 3D Modeling

At the core of the 3D modeling module is a stereo-based reconstruction pipeline. In particular, it is composed of *image mask estimation*, *depth map computation* and *depth map filtering*. In the following, each of these steps is discussed in more detail. Finally, the filtered depth map is back projected to 3D, colored with respect to the reference image and merged with the current point cloud.

6.1. Image Mask Estimation

The task of the maintained image mask is twofold. First, it identifies pixels exhibiting sufficient material texture. This allows to avoid unnecessary computations which have no or negligible effect on the final 3D model and reduces potential noise. Second, it overcomes the generation of redundant points by excluding regions already covered by the current point cloud.

A texture-based mask is computed by reverting to the Shi-Tomasi measure used also at the visual tracking stage (see Section 5). The mask is obtained by thresholding the values at some $\lambda_{min} > 0$. In our implementation, we set $\lambda_{min} = 0.1$ and use patch windows of size 3×3 pixels. Additionally, another mask is estimated based on the coverage of the current point cloud. To this end, a sliding window, which contains a set of the recently included 3D points, is maintained. All points are projected onto the current image and a simple photometric criterion is evaluated. Note that points, that belong to parts of the scene not visible in the current view, are unlikely to have erroneous contribution to the computed coverage mask. The final image mask is obtained by fusing the estimated texture and coverage mask. Subsequent depth map computations are restricted to pixels within the mask.

6.2. Depth Map Computation

Multi-resolution scheme. We run binocular stereo by taking an incoming image as a reference view and matching it with an appropriate recent image in the provided series of keyframes. Instead of applying a classical technique based on estimating the optimal similarity score along respective epipolar lines, we adopt a multi-resolution scheme. The proposed approach involves downsampling the input images, estimating depths, and subsequently upgrading and refining the results by restricting computations to a suitable pixel-dependent range.

Similar to the visual tracking stage (see Section 5), we rely on computations at multiple pyramid resolutions. At each level $i \in \{0, \ldots, L\}$, respective images are obtained by halving the resolution of their versions at level $i-1$ in each dimension. Thereby, $i = 0$ contains the original images. Starting at the top of the pyramid, the multi-resolution approach estimates a depth map $D_i : \Omega_i \subset \mathbb{Z}^2 \rightarrow R \subset \mathbb{R}$ to each level i based on the image data at that level and the depth map from the consecutive level D_{i+1}. While exhaustive computations have to be performed for the highest level L, subsequent computational efforts can be reduced significantly by exploiting the previously obtained coarse result. In particular, we apply an update scheme based on the current downsampled pixel position and three appropriate neighbors. For example, for pixel $(x, y) \in \Omega_i$ with $x \bmod 2 = 1$ and $y \bmod 2 = 1$ (the remaining cases are handled analogously) we consider the following already pro-

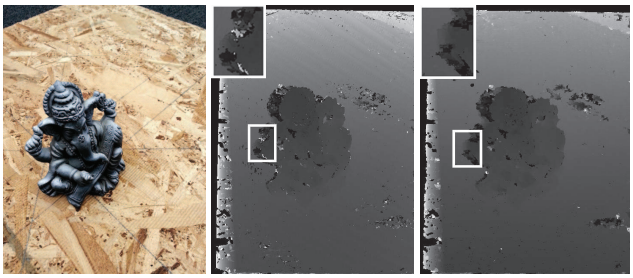

Figure 5. Single- vs. multi-resolution depth map estimation. *From left to right:* The reference image of a stereo pair, corresponding depth map estimated with a classical single-resolution winner-takes-all strategy and result obtained with the proposed multi-resolution scheme.

vided depth values

$$\begin{aligned} D^0_{i+1} &= D_{i+1}(x', y') \\ D^1_{i+1} &= D_{i+1}(x'+1, y') \\ D^2_{i+1} &= D_{i+1}(x'+1, y'+1) \\ D^3_{i+1} &= D_{i+1}(x', y'+1), \end{aligned} \tag{8}$$

where $(x', y') := (\lfloor x/2 \rfloor, \lfloor y/2 \rfloor) \in \Omega_{i+1}$. We estimate the depth $D_i(x, y)$ by searching an appropriate range given by the minimum and maximum value in $\{ D^l_{i+1} | l = 0, \ldots, 3 \}$. Thereby, depth values, that are not available due to boundary constraints or the maintained image mask, are omitted. In order to take into account uncertainties due to the coarse sampling at higher pyramid levels, we additionally include a small tolerance in the estimated ranges. As the uncertainty is expected to increase with increasing depth due to the larger jumps of the values from pixel to pixel, we use a tolerance parameter which is inversely proportional to the local depth D^0_{i+1}.

In our implementation, we used images of size 640×480 pixels and 3 resolution levels (i.e. $L = 2$). It should be noted that all estimated depth maps rely on a predefined range $R \subset \mathbb{R}$ which can be determined by analyzing the distribution of the sparse map constructed in the camera tracking module (see Section 5).

The multi-resolution scheme comes along with some important advantages. First, it entails significant efficiency benefits compared to traditional methods as epipolar line traversals at higher image resolutions are restricted to short segments. Second, when applying a winner-takes-all strategy, potential mismatches can be avoided due to the more robust depth estimates at low image resolution. Third, regions in the image, which belong to distant scene parts outside of the range of interest, can be discarded at the lowest resolution level and subsequent refinement operations can be avoided for them. To illustrate these aspects, we present a comparison between the classical single-resolution winner-takes-all strategy and the developed multi-resolution technique on an example image pair (see Fig. 5). While both results look genuinely similar, a closer look reveals that the proposed multi-resolution scheme confers a higher degree of robustness by producing less outliers. Note that for the above example a conservative depth range was used so as to capture the entire field of view. However, the real benefit from it becomes evident when comparing the respective runtimes. In practice, the multi-resolution approach is about 5 times faster than the single-resolution counterpart.

GPU acceleration. Despite the utilization of a multi-resolution scheme, the developed method for dense stereo is not efficient enough to meet the requirements of the application at hand. For this reason, we made use of the parallelization potential of the algorithm with a GPU implementation (based on GLSL ES), which reduces the overall runtime of the 3D modeling module to about 2-3 seconds per processed image. More concretely, we estimate depth maps at different pyramid levels in separate rendering passes. Thereby, some care should be taken due to the precision limitations of current mobile GPUs. We address this difficulty by using the sum of absolute differences (SAD) as a similarity measure in the matching process (over 5×5 image patches) and transferring triangulation operations to get the final depth estimates to the CPU.

Image pair selection. A crucial step in binocular stereo is the choice of an appropriate image pair. An ideal candidate pair should share a large common field of view, a small but not too small baseline and similar orientations. As we have an ordered image sequence, a straightforward methodology would be to match each incoming image with its predecessor. Yet, this strategy is suboptimal in some cases, for example when the user decides to move back and recapture certain parts of the scene. Instead, we propose to maintain a sliding window containing the last N_v provided keyframes ($N_v = 5$ in our implementation) and pick the one maximizing a suitable criterion for matching with the current view. For two cameras j and k this criterion is defined as

$$C(j,k) = \cos\theta^{jk}_{pose} \cdot \cos\theta^{jk}_{view} \cdot \cos\theta^{jk}_{up}, \tag{9}$$

where θ^{jk}_{pose} denotes the angle between the viewing rays of both cameras at the midpoint of the line segment connecting the mean depth range points along the camera principal rays, θ^{jk}_{view} is the angle between the principal rays and θ^{jk}_{up} is the angle between the up vectors of both cameras. Additionally, we impose the following constraints

$$5^\circ \leq \theta^{jk}_{pose} \leq 45^\circ, 0^\circ \leq \theta^{jk}_{view} \leq 45^\circ, 0^\circ \leq \theta^{jk}_{up} \leq 30^\circ$$

An input image is discarded and not processed if none of the images in the current sliding window satisfy those constraints with respect to it.

6.3. Depth Map Filtering

The final step in the proposed 3D modeling pipeline consists in filtering the estimated depth map. The applied procedure is inspired by [21] and is based on checking consistency over multiple views. In particular, a sliding window containing the last N_d depth maps is maintained. The depth value at each pixel of the current map is tested on agreement with the maps in the sliding window, i.e. it is warped to the corresponding views and compared with the values stored there. The depth is considered consistent if the estimated difference is within a certain tolerance for at least N_c views. In our implementation, we set $N_d = 5$ and $N_c = 2$. It is important to note that the unfiltered depth maps have to be maintained here because parts of the scene, not visible in the views included in the current sliding window, would never get the chance to be reconstructed otherwise. This simple but very powerful filtering procedure is able to remove virtually all outliers and build a clean 3D model.

7. Experimental Results

All experiments were conducted on a Samsung Galaxy SIII I9300GT with Samsung Exynos 4 quad core CPU and ARM Mali-400 MP4 GPU and processed in real time.

The first set represents a typical use case for mobile 3D reconstruction: non-movable objects for which no 3D geometry exists yet. To this end, multiple objects were captured from the collection of a museum. Fig. 6 shows the generated model of a tribal mask which was created interactively on-device during the normal opening hours of the museum. Fig. 7 depicts an additional object from the same collection: an ancient Shakyamuni Buddha statue. Both results were obtained under normal exhibition conditions. They illustrate the suitability for a number of practical cases where 3D objects cannot be simply taken to a full-fledged 3D scanning device.

To evaluate the generality of our approach, we complete the evaluation with two additional scenarios that were not envisioned initially: outdoor environments (see Fig. 8) and human faces (see Fig. 9).

See the accompanying video for additional reconstruction results and a demo of the developed system in action.

8. Conclusion

We presented the first interactive on-device system for dense stereo-based 3D reconstruction on mobile phones. In order to address the major challenges posed by the underlying hardware limitations and to meet the robustness and efficiency requirements of the application, we integrated multiple novel solutions. In particular, we explored the capabilities of the inertial sensors available on modern mobile devices to improve the resilience of the camera tracking process to rapid motions and to automatically capture keyframes when the phone is static. Additionally, we

Figure 6. Photo and front and right view of the reconstructed 3D model of a 0.5 m tall African tribal mask.

Figure 7. Photo and front and left views of reconstructed 3D model of a 1.6 m tall Shakyamuni Buddha statue.

Figure 8. Photo and reconstructed 3D model of a building facade captured at street-level.

Figure 9. Front and left view of a reconstructed 3D model of a human face, including a corresponding photo of the test person.

showed how this sensor information can be exploited to derive the metric measures of the captured scene. Moreover, we proposed an efficient and accurate method for binocular stereo based on a multi-resolution scheme. The performance of the system was demonstrated on various indoor and outdoor scenes.

Acknowledgments

We thank Amael Delaunoy for technical support and useful comments. Tobias Nägeli provided support and input on inertial attitude estimation. This work is funded by the ETH Zurich Postdoctoral Fellowship Program, the Marie Curie Actions for People COFUND Program and ERC grant no. 210806.

References

[1] P. Corke, J. Lobo, and J. Dias. An introduction to inertial and visual sensing. *The Intl. Journal of Robotics*, 2007. 2

[2] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 5

[3] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. *ISMAR*, pages 225–234, 2007. 2, 4, 5

[4] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. *ISMAR*, pages 83–86, 2009. 2

[5] M. Li and A. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *International Journal of Robotics Research*, 32(6):690–711, May 2013. 2

[6] J. Lobo and J. Dias. Inertial sensed ego-motion for 3d vision. *Journal of Robotic Systems*, 21(1):3–12, 2004. 2

[7] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327, 2011. 1, 2

[8] R. A. Newcombe et al. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011. 1

[9] Q. Pan, C. Arth, E. Rosten, G. Reitmayr, and T. Drummond. Rapid scene reconstruction on mobile phones from panoramic images. In *ISMAR*, pages 55–64, 2011. 1, 2

[10] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3):207–232, 2004. 2

[11] M. Pollefeys et al. Detailed real-time urban 3d reconstruction from video. *Int. J. Comput. Vision*, 78(2-3):143–167, 2008. 2

[12] L. Porzi, E. Ricci, T. Ciarfuglia, and M. Zanin. Visual-inertial tracking on android for augmented reality applications. In *IEEE Workshop on Environmental Energy and Structural Monitoring Systems (EESMS)*, pages 35–41, 2012. 2

[13] V. A. Prisacariu, O. Kaehler, D. Murray, and I. Reid. Simultaneous 3D tracking and reconstruction on a mobile phone. In *ISMAR*, 2013. 1, 2

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011. 4

[15] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. In *SIGGRAPH*, pages 438–446, New York, NY, USA, 2002. ACM. 1

[16] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, Apr. 2002. 2

[17] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 519–528, 2006. 2

[18] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994. 5

[19] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, USA, 2008. 2

[20] J. Stuehmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Pattern Recognition (Proc. DAGM)*, pages 11–20, 2010. 2

[21] E. Tola, C. Strecha, and P. Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Mach. Vision Appl.*, 23(5):903–920, 2012. 7

[22] G. Vogiatzis and C. Hernndez. Video-based, real-time multi-view stereo. *Image Vision Comput.*, pages 434–441, 2011. 2

[23] S. Weiss and R. Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *ICRA*, pages 4531–4537, 2011. 2, 4

[24] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1450–1457, 2012. 2