3DCapture: 3D Reconstruction for a Smartphone

Oleg Muratov Yury Slynko Vitaly Chernov Maria Lyubimtseva Artem Shamsuarov
Victor Bucha
Samsung R&D Institute RUS
{o.muratov, y.slynko, v.chernov, l.maria, v.bucha}@samsung.com

Abstract

We propose a method of reconstruction of 3D representation (a mesh with a texture) of an object on a smartphone with a monocular camera. The reconstruction consists of two parts – real-time scanning around the object and post-processing. At the scanning stage IMU sensors data are acquired along with tracks of features in video. A special care is taken to comply with 360° scan requirement. All these data are used to build a camera trajectory using bundle adjustment techniques after scanning is completed. This trajectory is used in calculation of depth maps, which then are used to construct a polygonal mesh with overlaid textures. The proposed method ensures tracking at 30 fps on a modern smartphone while the post-processing part is completed within 1 minute using an OpenCL compatible mobile GPU. In addition, we show that with a few modifications this algorithm can be adopted for human face reconstruction.

1. Introduction

Many recent works [19, 14] have shown that 3D reconstruction on a smartphone became a reality. However, most solutions are still quite computationally heavy and do not produce a high quality model. These methods generate dense point cloud or 3D color volume, which usually considered as insufficient for consumer product. Another disadvantage of most solutions is a tricky initialization procedure [13].

In this paper, we propose a method of reconstruction of a 3D model of an object as a mesh with a texture. High frame rate is crucial for stable tracking (due to small pose changes between sequential frames at high frame rate) thus we are using a lightweight 2D feature points tracker coupled with IMU sensors integration. This approach also allows us to exclude an initialization step. Then points tracks and IMU data are used in a novel structure from motion algorithm to calculate a trajectory of a camera. During 360° scanning, points are visible for a short period of time and strong occlusion occurs making pose estimation difficult. But we can rely on the fact that the end of the trajectory lies close to its start thus a loop closure techniques is used to improve structure from motion accuracy.

Given a camera poses and keyframes it is straightforward to create depth maps and fuse them into a voxel volume. This volume is converted into a mesh and complemented with textures. Thanks to GPGPU computations for depth processing stages our implementation generates a high resolution mesh within 1 minute on a smartphone.

Our key contributions can be summarized as:

- A novel structure-from-motion algorithm based on robust real-time feature tracking, IMU data integration and appropriate offline bundle adjustment.
- An optimized pipeline for reconstructing a 3D representation of an object as a mesh with a texture completely on a mobile phone.

1.1. Related Work

The current work deals with monocular 3D reconstruction, which has been addressed by a great number of works. However, most of these approaches are inapplicable for mobile device use due to limited computational resources. Thereby, below we will focus solely on works that address mobile device application of 3D reconstruction algorithms.

One of the first works on 3D reconstruction from mobile phone is [19]. It is based on the well known PTAM algorithm [6] complemented with inertial data integration. Keyframes with poses produced by SLAM system are used for depth map computation in real time, which are fused into a dense 3D color point cloud. A more advanced approach has been described in [14]. It utilizes direct model-based image alignment for camera pose estimation. At the same time the reconstructed volumetric model is continuously updated with new depth measurements. Another approach has been presented in [15]. It performs simultaneous visual-inertial shape-based tracking and 3D shape estimation.

All above mentioned methods use online[1] tracking and mapping, which is a computationally heavy task. This lead to two significant drawbacks of such approaches. Firstly, due to limited CPU/GPU/memory resources they compromise reconstruction quality to achieve real-time performance. Secondly, such systems require accurate and smooth scanning that is hard to achieve for an untrained user.

As opposed to these methods, a structure from motion (SfM) approach can be used for the same task with no need for real-time performance. For instance, OpenMVG[12] framework is used in [18]. However, in order to recover camera trajectory such approaches usually use local feature matching that results in a quite long offline processing. In many cases, as in [1, 21], these tasks are done using cloud-based processing.

Our approach is close to classic SfM approaches. However, we propose an efficient motion estimation algorithm that combines optical flow and inertial data that allows faster runtime and higher accuracy as compared to pure SfM approaches, such as [18]. Moreover, to the best of our knowledge this is the first work that presents a complete 3D reconstruction pipeline (from capture to textured mesh) for mobile devices.

1.2. Structure of the Paper

This paper is organized as follows: we present an overview of the proposed 3D scanning and reconstruction method in Sec. 2. The motion estimation method and the 3D reconstruction pipeline are described in Sec. 3 and Sec. 4 respectively. Finally, we evaluate the proposed solution in Sec. 5.

2. System Overview

A block-wise diagram of our approach is shown in Fig. 1. We start from capturing a video during which a user is asked to make a loop around an object of interest. During this scanning IMU measurements are collected and visual feature tracking is performed.

After capturing is completed, keyframes with tracked features and IMU data are used to estimate camera trajectory and scene structure. Next, the keyframes with poses and sparse scene structure are passed to the depth map estimation module, where a depth map is computed for each keyframe. These depth maps are fused robustly into a single voxel volume. After that a 3D mesh is extracted from this volume. Finally, given the keyframes with poses and the 3D mesh, a texture is generated.

The result of the proposed method is the 3D textured mesh model of an object of interest. The proposed method tracks online at 30 fps on a modern smartphone while the offline part takes less then a minute using an OpenCL compatible mobile GPU.

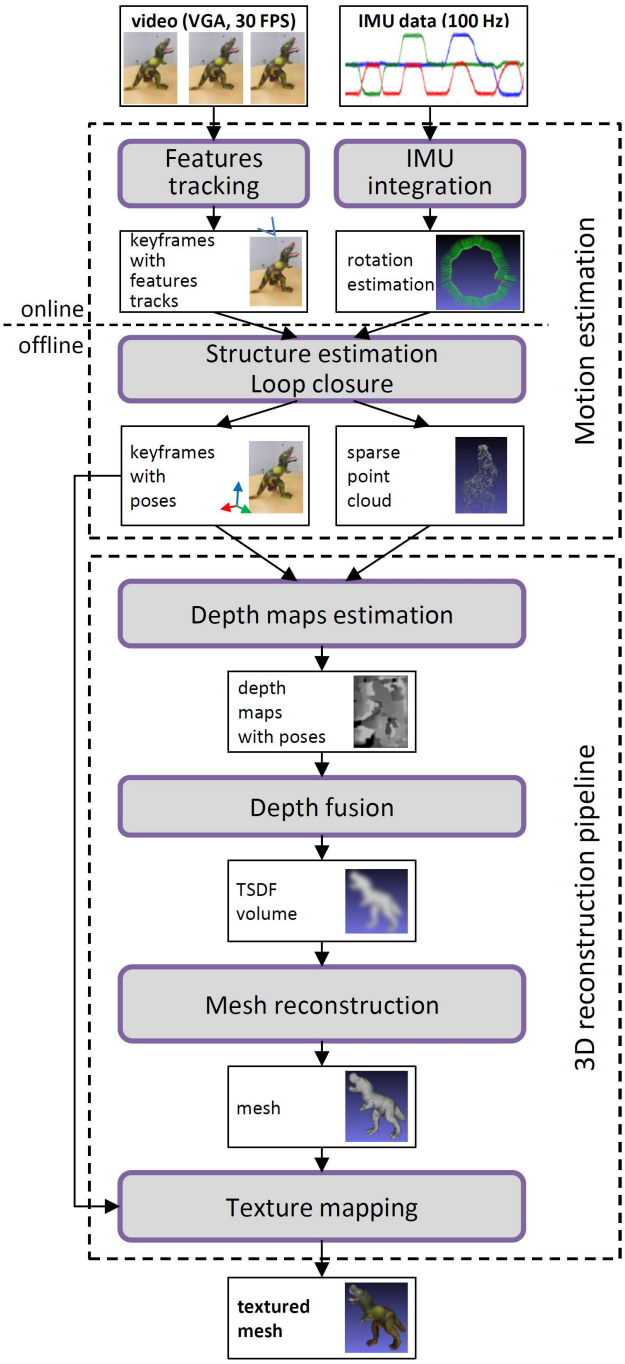


Figure 1. System Overview

In the following sections each step of algorithm is described in details.

[1]Here and below, online and offline terms correspond to a time period during and after capturing respectfully.

3. Motion estimation

A motion estimation module calculates a trajectory of a camera. As typical for such problems it also calculates some scene geometry information (a sparse point cloud).

This module is divided into online and offline parts. Inertial rotation estimation and feature tracking parts gather information online in two parallel threads while structure estimation and loop closure is done offline.

Unlike traditional SLAM systems there is no tight coupling of tracking and mapping parts: feature tracking is not dependent on the results of structure estimation, the latter do not need to meet real-time requirements and can be performed after scene capturing has been finished. This leads to three key properties of the proposed method:

1. There is no initialization step in contrast to a typical SLAM system.
2. Online part is quite lightweight and performance does not degrade with time.
3. Lack of feedback loop between tracking and mapping parts makes tracking very stable since outliers do not affect online parts of an algorithm.

The inertial rotation estimation thread performs gyroscope measurements integration to provide rough rotation estimation. The gyroscope sensor is sampled at the highest available rate, it is 100 Hz in our case. For the sake of efficiency rotation data are represented in a quaternion form. New measurements are applied to the current rotation estimate using a first order quaternion integration [20]. Here we do not compensate for the possible drift due to bias. In our computation we use only relative rotation between keyframes that are very close in time that makes effect of this drift negligible.

The feature tracking thread selects keyframes for further processing and establishes visual correspondences between them. In order to get correspondences we select a set of features in keyframes. As the features we use grayscale 8×8 square patches taken at FAST corners [16]. We align them in frames with 2D Lucas-Kanade tracker [2] using position of patch at previous frame as initial guess. For the sake of efficiency alignment is performed in inverse-compositional manner on a pyramidal image in a coarse-to-fine fashion.

For better robustness against outliers a bidirectional alignment is performed: a patch from a keyframe is aligned in the current frame, and then this aligned patch from the current frame is aligned back to the keyframe. If such bidirectional alignment does not return to the same position in the keyframe from where it is started, then alignment is treated as failed.

It is computationally expensive to track all detected features, thereby only a subset of detected features is selected for tracking. A grid-based filtering is applied: an image is divided into cells, and the feature with the highest Shi-Tomasi response [17] is taken from each cell. Such a filtering ensures feature points are evenly distributed over an image. This minimizes a possible gauge freedom effect.

In a scenario of a circular loop motion a viewpoint changes rapidly such that features are observed for a short time, and their 2D projections appearance changes a lot. To tackle with feature disappearance new keyframes are initialized based on camera rotation obtained from gyroscope data. Empirically we found that generating a new keyframe every 3° results in best motion estimation accuracy. From each keyframe a new set of features is extracted.

In addition, features are updated each time a new keyframe is generated. If given feature is aligned in the new keyframe, its aligned position and patch from this keyframe is used for alignment in next frames. This ensures that majority of features are observed for over 10 keyframes resulting in dense connectivity between frames while preserving negligible features drift.

The structure estimation part receives features tracks and keyframes with associated rotation and estimates camera trajectories along with a 3D structure of the scene. All features which were observed in less than three keyframes are filtered out.

The main idea of the structure estimation algorithm is to use rough rotation transform between frames calculated from gyroscope measurement and reduce the pose estimation problem (which is a 6 DOF problem) to finding a corresponding translation (which is a 3 DOF problem). The core of the algorithm is a multiple view matrix of a point [11]:

$$M_p \doteq \begin{bmatrix} \widehat{\mathbf{x}_2^j} R_2 \mathbf{x}_1^j & \widehat{\mathbf{x}_2^j} T_2 \\ \widehat{\mathbf{x}_3^j} R_3 \mathbf{x}_1^j & \widehat{\mathbf{x}_3^j} T_3 \\ \vdots & \vdots \\ \widehat{\mathbf{x}_m^j} R_m \mathbf{x}_1^j & \widehat{\mathbf{x}_m^j} T_m \end{bmatrix} \in \mathbb{R}^{3(m-1)\times 2}, \qquad (1)$$

where $\mathbf{x}_i^j$ is normalized homogeneous coordinates of a point p_j in a frame C_i, $\widehat{\mathbf{x}}$ is a cross product operator, R_i and T_i are rotation and translation of the frame C_i respectfully. From the rank condition of M_p it follows:

$$\widehat{\mathbf{x}_i^j} R_i \mathbf{x}_1^j + \alpha^j \widehat{\mathbf{x}_i^j} T_i = 0, \qquad (2)$$

where term α^j is the inverse depth of a point p_j with respect to the first frame. These equations are stacked for each point to form a system of equations. Given initial estimate for rotation and depth this problem takes a normal form of $A\mathbf{x} = \mathbf{b}$, which can be solved efficiently using Least-Squares with respect to T_i. And vice versa, inverse depth is estimated using multiple-view constraint equation:

$$\alpha^j = -\frac{\sum_{i=2}^{m}(\widehat{\mathbf{x}_i^j T_i})^T \widehat{\mathbf{x}_i^j} R_i \mathbf{x}_1^j}{\sum_{i=2}^{m} \|\widehat{\mathbf{x}_i^j}\|^2}, \quad (3)$$

with m denoting the number of frames where a point p_j has measurements.

For the first pair of frames there is no prior information on translation and depth data. In order to initialize structure we first set inverse depth values for all points to 1, thus making an assumption of a plane in the front of the camera. Then an iterative procedure is applied. At each iteration we estimate translation T_i and inverse depths α^j for all points except a reference one. This process stops when reprojection error becomes below a threshold. The inverse depth value of the reference point is fixed to 1 for the entire structure estimation process, thus defining a scale of the whole scene. There can be different strategies to select this reference point; in our implementation we pick a point with a projection nearest to the center of the frame.

For consecutive frames there is no need to perform "cold start", translation can be computed based on points which already have inverse depth estimate. Then, a depth is updated using equation (3). Note that this update includes not only points without depth value, but all the points visible in the current frame.

Next, we refine our estimates by sequentially performing motion-only and structure-only bundle adjustment optimizations, which minimize reprojection residuals using Gauss-Newton algorithm. This is necessary in order to compensate for a possible rotation drift due to gyroscope sensor bias. On average it requires only 1-2 iterations till the system converges.

After all visual correspondences have been processed **the loop closure step** is performed. This step takes advantage of our prior knowledge that the trajectory contains a loop point due to a circular-like shape. From the first keyframe we extract BRISK [10] features computed at FAST corners and seek for loop closure points among other keyframes. For efficiency reasons this comparison is done only for keyframes that are within 15° of angular distance to the first keyframe and have no common points with it. Once a good set of correspondences is found it is augmented into the existing scene structure and the pose transform from the first to the loop closure keyframe is computed. In order to incorporate this information a global bundle adjustment is performed using g^{2}o framework [8].

4. 3D Reconstruction Pipeline

4.1. Depth Map Estimation

The keyframes with poses are passed as an input to a depth map estimation algorithm. The result of this module is a set of the depth maps (with corresponding poses).

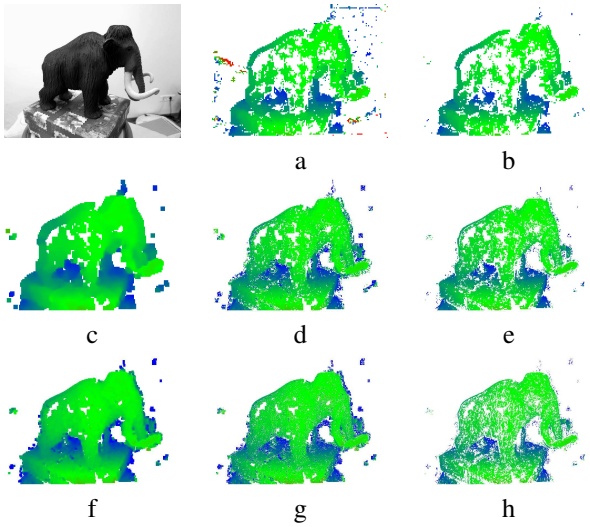


Figure 2. Coarse-to-fine depth estimation: depth map with ambiguity filtering on level 0 (a), the same after left-right consistency filtering (b); upscale from level 0 to 1 (c), depth map with ambiguity filtering on level 1 (d), the same after left-right consistency filtering on level 1 (e); upscale from level 1 to 2 (f), depth map with ambiguity filtering on level 2 (g), the same after left-right consistency filtering on level 2 (h).

This algorithm is based on a planesweep approach described in [3]. Computational simplicity of this method ensures fast calculation even on a mobile device. At the same time, raw depth measurements without an excessive regularization help to preserve fine details in a 3D model. This happens because our variational depth fusion method (see Section 4.2) suppresses the impulse-like noise of the simple planesweep approach. However, it can not deal with large patches of wrong depth information that usually come from methods with strong regularization. For this reason we omit a regularization step in contrast to [3].

Depth estimation is done in a coarse-to-fine pyramidal scheme with three levels of the pyramid. On the coarsest pyramid level only a small number of pixels must be processed. This allows us to use more accurate settings for depth estimation on this level without sacrificing runtime. For instance, we use 5×5 versus 3×3 search window and perform image rectification. For the upper levels of the image pyramid, rectification is omitted for efficiency reasons. In addition, for better accuracy and faster convergence search range is calculated using the sparse point cloud created during motion estimation.

Depth filtering is applied to each pyramid level. Fusion methods can usually handle well missing data by interpolation or propagation from existing values, but are more sensitive to heavy outliers. Thus, we leave only the depth values which are accurate with high confidence.

We propose two stages of depth filtering.

- *Photometric ambiguity.* The depth outliers can be efficiently filtered out by analyzing ratios of the costs to its minimal value for each pixel (see Fig. 2(a)). When a texture is absent or ambiguous (periodic along the epipolar line) many costs will have ratios around 1. This allows to filter these ambiguities. An example of the resulting depth maps with the photometric ambiguity filtering applied is shown in Fig. 2(a)(d)(g).

- *Left-right consistency.* The left-right check is done by analyzing the consistency of both depth maps for the left and right images in the stereo pair. The consistency is determined by checking re-projection errors for each pixel using depth values from both depth maps. An examples of the depth maps after the left-right consistency check are shown in Fig. 2(b)(e)(h).

The depth filtering stage significantly reduces a number of pixels to be processed on the next pyramid level. It is especially important because the finest pyramid levels processing is much slower than coarser ones. The proposed depth estimation algorithm allows a very efficient parallel implementation on a graphical processor (GPU). Memory consumption can also be reduced because there is no need to store a full cost-volume in a global memory. An image can be processed in small regions, and for each of them the matching cost values can be stored in a local memory.

4.2. Depth Fusion

The depth fusion module fuses (combines) all depth maps into a volumetric representation of the object taking into account their poses. We follow approach described in [22, 3] for variational fusion of depth maps using truncated signed distance function (TSDF), which is implemented in a coarse-to-fine manner.

The reconstruction takes place in the volume of interest (VOI), which is automatically placed around the captured object based on the sparse point cloud available after tracking stage. On the finer pyramid level the result of optimization procedure is used as an initial guess. This allows us to speedup runtime by reducing the number of iterations required for convergence at the optimization stage and improve quality by quickly propagating measurements to unseen space on coarser levels and thus reducing artifacts which typically appear outside of a captured area.

In addition, we improve spatial resolution of the final model by re-adjusting VOI after analyzing results of the coarse level. Often an object of interest is placed on a table or another planar support structure (e.g. box). First, we examine if there is a horizontal plane at the bottom of the captured scene. This is done by detecting a plane in the dense point cloud using SVD and checking that all camera rays are casted from above on that plane. This allows alignment of VOI with the plane and avoid wasting space for volume

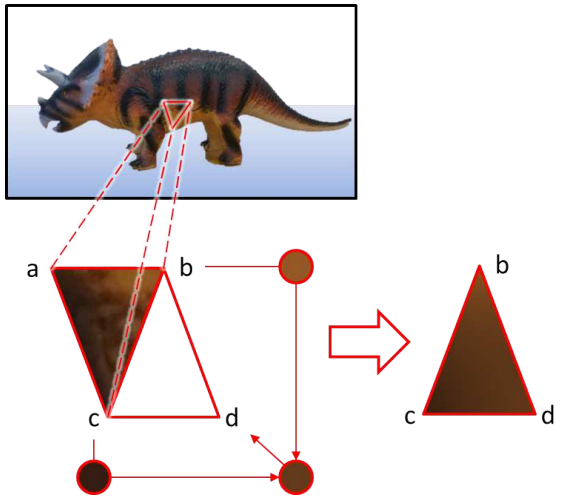


Figure 3. Vertex color interpolation. Here two faces are shown: *a-b-c* (textured) and *b-c-d* (not textured yet). First, color to vertices *b* and *c* is assigned. Color value for vertex *d* is obtained by averaging colors of vertices *b* and *a*. Texture for face *b-c-d* is obtained by interpolating color values of vertices *b, c* and *d*.

below it. After processing coarse pyramid level rough volumetric representation of the scene is used to refine information about space that is empty or occluded by the support structure. This is done by examining whether TSDF values are close to 1, which represents a volume between a camera and the surface. VOI boundaries are automatically placed more tightly around the object, excluding processing of the useless area.

Similar to [3] we use first-order primal-dual algorithm for minimization of the energy function on each level of pyramid. The result is a TSDF volume of the captured object. Our implementation uses 3 levels of pyramid with 128^3 volume on the finest level. Measurement integration and energy minimization is done on GPU with OpenCL kernels.

4.3. Mesh Construction and Simplification

With the computed TSDF representation a 3D polygonal mesh can be reconstructed in the two following steps.

Octree-based TSDF representation. Since the computational complexity of texture mapping procedure is linearly dependent on a number of the 3D mesh polygons, we use octrees representation of the TSDF volume. For the most models, a maximum octree depth value equal to 7 limits the complexity of the 3D model by 20-30 thousand polygons.

Isosurface extraction. The 3D mesh is reconstructed using an unconstrained isosurface extraction on arbitrary octrees approach as it is described in [5].

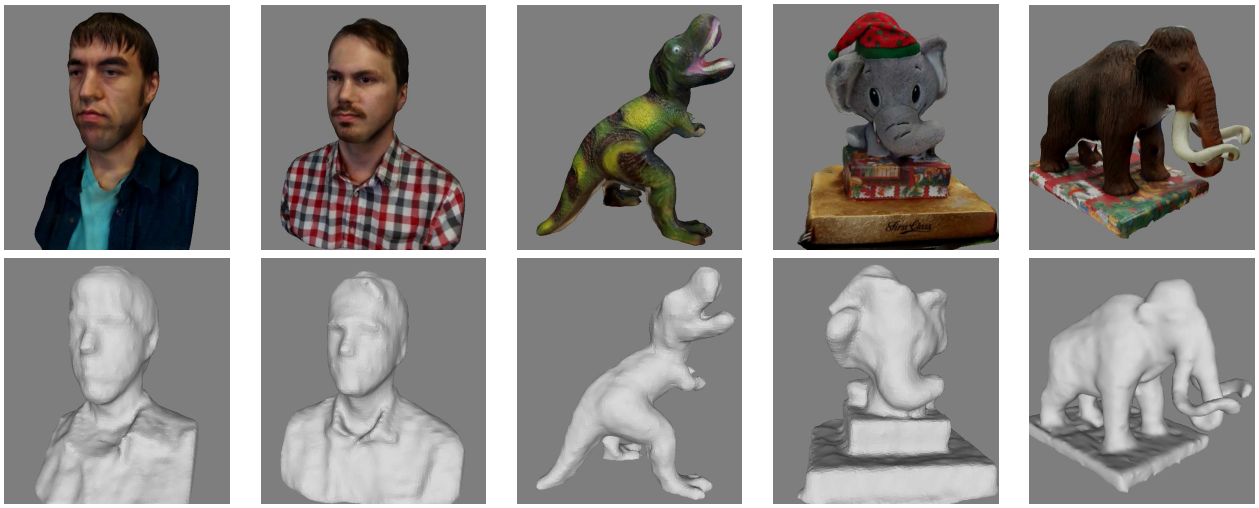
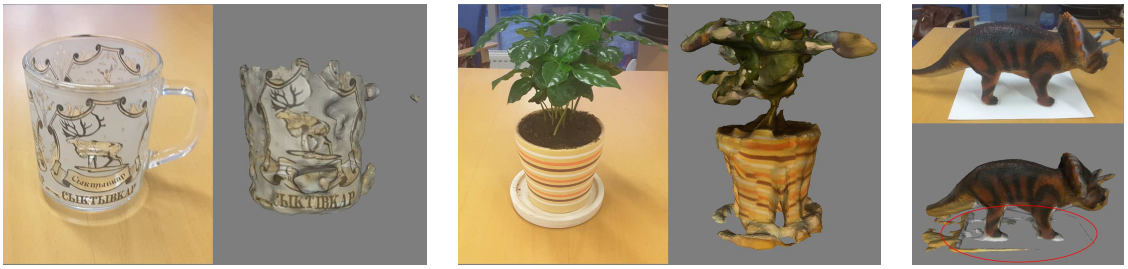


Figure 4. Sample reconstruction results.



a b c

Figure 5. Method limitations: transparent and specular object (a), plane (2D) details (b), objects with no texture (c).

4.4. Texture Mapping

The reconstructed mesh and keyframes with poses are used as an input of the texture mapping algorithm which builds a textured mesh. First it processed visible faces, then creates texture for invisible ones.

Seamless texturing. Each visible face is textured by means of projection to one of camera images. To achieve best result special care is paid to avoid seams between texture patches from different keyframes. This is done in two stages following an algorithms described in [9].

First, for each visible face a camera which will be used for texturing is selected by means of solving Markov random field (MRF) energy minimization problem. Objective function consists of two terms that forcing each face to choose individually "best" camera at the same time forming seamless texture between adjacent faces. Second, textures are adjusted by adding special leveling function to them to minimize color discontinuities on seams, left after the first step.

Hole filling. There can be some faces that are invisible from any camera. Hence there are some not textured areas (holes) in the mesh at this stage. To fill these holes we propagate color information from textured faces to adjacent untextured faces through simple vertex color interpolation.

First, all invisible vertices are added into a processing set. Then a color of each vertex from the set is computed by averaging colors of all visible adjacent vertices. Color for a visible vertex is picked from the same camera image as for texturing faces which contain this vertex. If there are several faces containing this vertex that are labeled differently, either of them is chosen. All colored vertices are marked as visible and deleted from the processing set. The process stops when the set is empty. The vertex color interpolation algorithm is shown in Fig. 3.

4.5. Face Reconstruction

In case of capturing human face we apply additional steps for better reconstruction quality. First of all, we use face detection on the first captured frame in order to decide if we need to take additional care during reconstruction. For face detection we utilize 3D facial motion capture algorithm [4]. This algorithm provides information such as 3D position of the head, face contours, precise eyes and mouth location, gaze direction and blinking information.

During structure estimation we use eyes location information and sparse scene structure in order to get metric estimate of scene scale. This allows us to use of metric thresh-

olds in further processing providing better reconstruction results. In the depth map estimation algorithm the face area is used in order to eliminate depth outliers. Eyes location is also used in order to properly place and align VOI during depth fusion. For better texture mapping we use eyes and mouth textures from the same camera view (to ensure consistent eyes' view direction), which is chosen to be the most frontal camera view at the same time avoiding views where blinking has been detected.

5. Experimental Results

For the evaluation we run the proposed algorithm on Samsung Galaxy Tab S2 tablet with Samsung Exynos 5433 SoC featuring four core CPU and ARM Mali-T760 GPU. In our implementation we retrieve HD images from the camera. For motion and depth processing we downsample images to VGA resolution. For texture mapping image resolution is critical for the quality and HD images are used as sources for textures. Timing per module is provided in Tab. 1.

In addition, we performed qualitative comparison of the proposed solution to the prior art. For this evaluation we used *Church* dataset provided by [19]. On Fig. 6 we show reconstruction quality comparison with algorithms described in [19, 7], [14] and [18]. It is evident that the proposed algorithm generates less noisy 3D model. It is worth mention that our and [18] solutions reconstruct a mesh while other considered methods produce a point cloud or SDF. Time of mesh generation (after capturing is finished) is within 1 minute for our approach and about 20-30 minutes for [18].

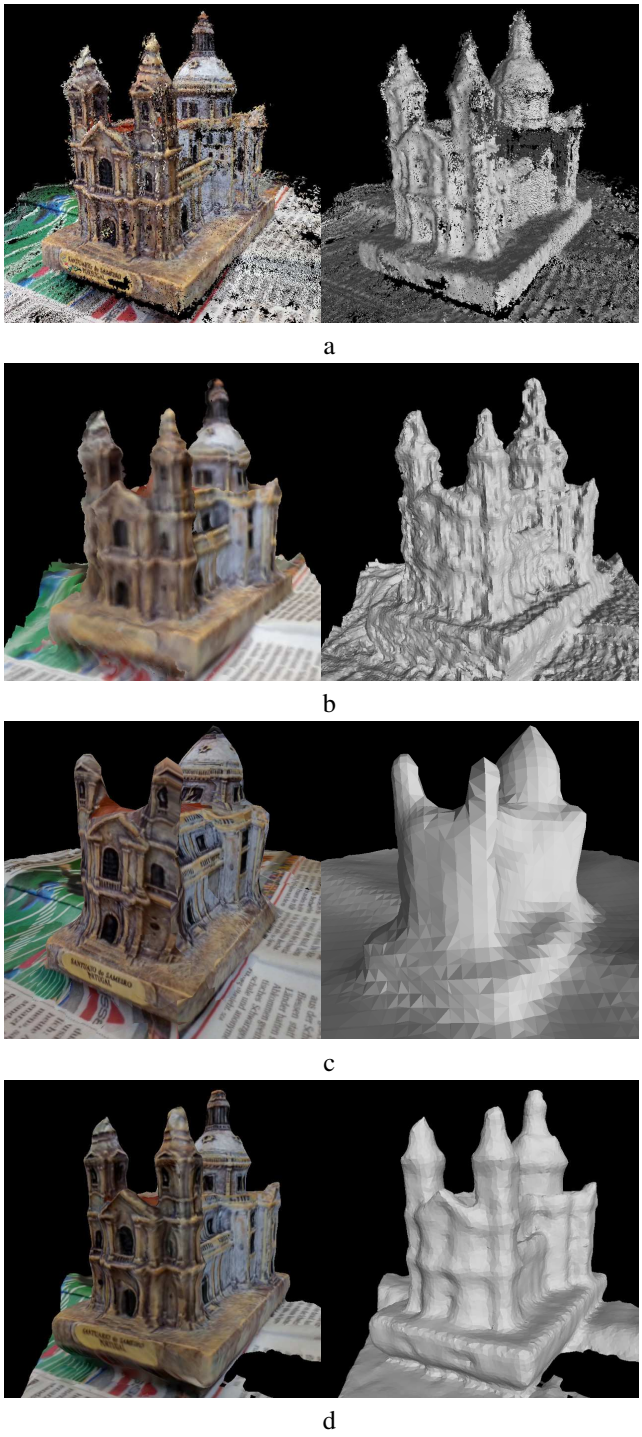
Some sample reconstruction results on more challenging datasets are shown in Fig. 4. The objects were captured indoor, under normal illumination conditions. The scanning path was a closed loop (360°) around an object. It is apparent the the proposed method can cope with quite small details in high level of occlusions.

Some limitations of our method are presented in Fig. 5. These limitation are quite common for monocular reconstruction: specular, transparent or glossy objects, objects with no texture and narrow planes (2D objects).

360° scanning is a difficult case for tracking systems and usually requires an accurate and smooth camera motion. In contrast, the proposed method allows even an untrained user to track successfully.

6. Conclusions and Future Work

We presented the world-first method of reconstruction of an object as a high-quality textured polygonal 3D mesh done completely on a mobile device with a monocular camera. Using of high-precision tracking system optimized for 360° scanning, IMU sensors integration thread, dedicated



a

b

c

d

Figure 6. Reconstruction results on *Church* dataset [19]: Kolev et. al. [7] (a), MobileFusion [14] (b), SCANN3D [18] (c) and our (d). Left and right images rendered using diffused and direct light respectfully. Vertex representation of MobileFusion results was obtained by applying *isosurface* function from Matlab to raw SDF volume.

Table 1. Timing by modules for Samsung Galaxy Tab S2. On average during scanning 120 keyframes are generated.

Module	Performance
scanning	30fps / 100Hz
structure estimation / loop closure	25s
depth maps	12s
mesh generation (inc. depth fusion)	8s
texture mapping	10s

camera pose estimation method based on bundle adjustment techniques, robust depth map fusion algorithm and accurate texture mapping method ensures a high quality 3D model. Overall processing time is within 1 minute due to parallelization of all critical algorithm parts using OpenCL framework and execution on mobile device GPU. The performance of the proposed solution was demonstrated on general object and human face cases.

References

[1] Autodesk, Inc. *123D Catch*. www.123dapp.com/catch. 2

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004. 3

[3] G. Graber. Realtime 3D reconstruction. Master's thesis, Graz, Austria, March 2011. 4, 5

[4] Y. Hwang, J.-B. Kim, X. Feng, W.-C. Bang, T. Rhee, J. D. Kim, and C. Kim. Markerless 3D facial motion capture system. In *IS&T/SPIE Electronic Imaging*, pages 828–909. International Society for Optics and Photonics, 2012. 6

[5] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe. Unconstrained isosurface extraction on arbitrary octrees. In *Symposium on Geometry Processing*, volume 7, 2007. 5

[6] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality (ISMAR)*, 2007. 1

[7] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning mobile phones into 3D scanners. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3946–3953. IEEE, 2014. 7

[8] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2011. 4

[9] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007. 6

[10] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011. 4

[11] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, 2003. 3

[12] P. Moulon, P. Monasse, R. Marlet, et al. OpenMVG. An open multiple view geometry library. https://github.com/openMVG/openMVG. 2

[13] A. Mulloni, M. Ramachandran, G. Reitmayr, D. Wagner, R. Grasset, and S. Diaz. User friendly SLAM initialization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 153–162. IEEE, 2013. 1

[14] P. Ondrúška, P. Kohli, and S. Izadi. MobileFusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *DOI 10.1109/TVCG.2015.2459902, IEEE Transactions on Visualization and Computer Graphics*, 2015. 1, 7

[15] V. A. Prisacariu, O. Kahler, D. W. Murray, and I. D. Reid. Simultaneous 3D tracking and reconstruction on a mobile phone. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 89–98. IEEE, 2013. 1

[16] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE, 2005. 3

[17] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 2007. CVPR'94. IEEE Conference on*. IEEE, 1994. 3

[18] SmartMobileVision Scann3D (candidate/3.2.1-4223). http://www.smartmobilevision.com. 2, 7

[19] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3D reconstruction on mobile phones. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 65–72. IEEE, 2013. 1, 7

[20] N. Trawny and S. I. Roumeliotis. Indirect Kalman filter for 3D attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2, 2005. 3

[21] TRNIO. *TRNIO*, 2015. http://www.trnio.com. 2

[22] C. Zach. Fast and high quality fusion of depth maps. In *Proceedings of the international symposium on 3D data processing, visualization and transmission (3DPVT)*, volume 1. Citeseer, 2008. 5