3D Modeling on the Go: Interactive 3D Reconstruction of Large-Scale Scenes on Mobile Devices

Thomas Schöps Torsten Sattler Christian Häne Marc Pollefeys
ETH Zurich, Switzerland
{thomas.schoeps, torsten.sattler, christian.haene, marc.pollefeys}@inf.ethz.ch

Abstract

This paper presents a system for 3D reconstruction of large-scale outdoor scenes based on monocular motion stereo. Ours is the first such system to run at interactive frame rates on a mobile device (Google Project Tango Tablet), thus allowing a user to reconstruct scenes "on the go" by simply walking around them. We utilize the device's GPU to compute depth maps using plane sweep stereo. We then fuse the depth maps into a global model of the environment represented as a truncated signed distance function in a spatially hashed voxel grid. We observe that in contrast to reconstructing objects in a small volume of interest, or using the near outlier-free data provided by depth sensors, one can rely less on free-space measurements for suppressing outliers in unbounded large-scale scenes. Consequently, we propose a set of simple filtering operations to remove unreliable depth estimates and experimentally demonstrate the benefit of strongly filtering depth maps. We extensively evaluate the system with real as well as synthetic datasets.

1. Introduction

Obtaining accurate, large-scale, and dense 3D reconstructions of environments in (or close to) real-time is a core problem in 3D Computer Vision. Recently, multiple solutions to this problem have been proposed that run interactively and rely on active depth sensors, *e.g.*, Microsoft's Kinect or the depth camera integrated in Google's Project Tango devices. In such systems the user walks around with a hand-held device and reconstructs the scene, allowing the user to directly add data where it is needed. Active sensors are usually restricted to indoor use because of too strong background illumination by the sun, and have a limited depth range. This creates a strong need for passive, image based solutions to overcome these problems.

In this paper, we present a scalable system for dense 3D reconstruction of large outdoor scenes based on monocular motion stereo. Our approach achieves interactive frame rates on Google's Project Tango Tablet by utilizing the device's GPU; see Fig. 1 for an example of a reconstruction obtained interactively on a tablet. Thanks to the device's fisheye camera a large field-of-view is observed in each image, which significantly simplifies and speeds up the capturing of larger scenes compared to standard lenses. Our method follows a two-step approach. First, depth maps are computed from an input video stream via stereo matching over time. Subsequently, these depth maps are fused globally using volumetric depth map fusion.

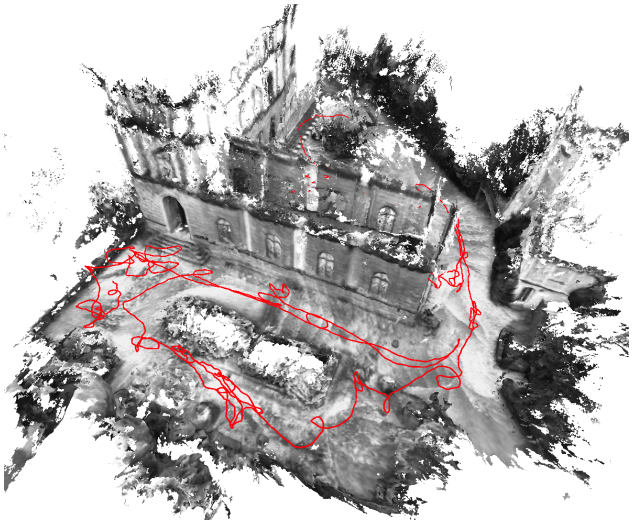

Figure 1. A model reconstructed by our system running at interactive frame rates on a Google Project Tango Tablet, with the camera trajectory shown in red. The user walked 373 meters around the building, which took about 12 minutes. The mesh is generated using motion stereo on a grayscale fisheye camera.

Compared to the output of active sensors, depth maps computed from images only contain significantly more noise and outliers. Additionally, due to for example textureless areas, depth can in general not be computed for all areas in an image. Depth measurements not only contain information about the surface position, but also about the free-space along the viewing ray. In the fusion step, this free-space information is used to keep the space in front of

objects free from unwanted surface patches reconstructed due to outliers in the input data. While for small-scale or indoor scenes usually the whole free-space can be observed through suitable choices of camera positions, this is not feasible for large-scale outdoor scenes. We tackle these problems by propagating depth data from frame to frame and identifying reliable measurements, which are fused locally using a Kalman filter. Outliers are discarded. The remaining depth observations are integrated into the volume.

This paper makes the following contributions. We present and evaluate a system for large-scale 3D reconstruction of outdoor scenes that runs at interactive frame rates on modern mobile devices. Our system uses monochrome fisheye images and thus does not come with the limitations of systems using active depth cameras, such as a restricted depth range and sensitivity to background illumination. We demonstrate the importance of filtering outliers in the depth maps and propose multiple filtering steps, which despite their computational simplicity significantly improve the reconstruction quality without affecting the real-time performance. An extensive experimental evaluation on both synthetic and real data shows that our system compares favorably against active sensor based approaches.

2. Related Work

Dense 3D modeling is a very well studied problem in computer vision. In this section, we discuss the most related works to ours. Therefore, we focus on efficient large-scale systems. Most of these systems first acquire depth maps from individual viewpoints and then fuse the depth data into one globally consistent 3D model. The systems largely differ in how the depth data is obtained and fused.

[25] generates 3D mesh models on a city scale out of street level video sequences in real-time by heavily relying on GPUs. Depth maps are computed using plane sweeping stereo [34], utilizing the predominant directions present in urban scenes [6]. Depth data from adjacent frames is then locally fused into higher quality depth maps [18].

[19] uses optical flow and regularization to compute high quality depth maps from video data of a monocular camera. DTAM [21] simultaneously uses the resulting dense reconstruction to robustly track the camera pose. Both approaches rely on the use of desktop GPUs to achive real-time performance.

Due to lack of texture it is often impossible to obtain reliable depth estimates for all pixels of a depth map. While [19, 21] employ regularization to fill in missing regions, [4,24,28,32] filter out unreliable points based on estimates of their uncertainty. [14, 31] present a 3D reconstruction system for mobile phones using monocular video input. For each keyframe, a depth map is estimated in a multi-resolution approach to gain speed and robustness. Unlike our system, their approaches do not achieve real-time performance but require a few seconds for processing.

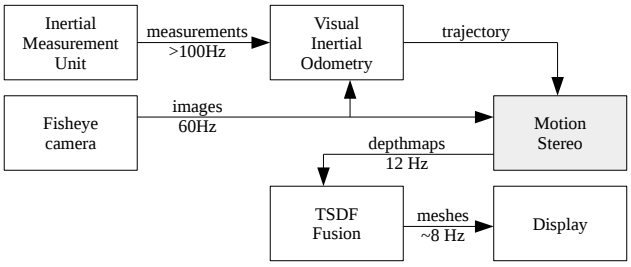

Figure 2. System diagram for the mobile real-time usecase.

Volumetric approaches fuse depth data by determining for each element of the space whether it is inside or outside of an object. A line of research using depth maps computed from images uses global regularization over the volume [15, 36] and as such the methods are generally non-interactive. Moreover, storing the whole space in terms of regular sized voxels is not feasible for large-scale systems. Subdividing the space into tetrahedrons based on depth data leads to an adaptive representation of the space [10] and allows for detailed reconstruction of whole cities or mountains from aerial images. Due to global visibility optimization and surface mesh refinement, this approach cannot operate in an incremental fashion, rendering it unsuitable for interactive 3D reconstruction. Interactive methods are feasible when only using a sparse set of 3D points and carving tetrahedrons [12, 23, 35].

With the availability of GPUs and structured light sensors, *i.e.*, RGBD (color and depth) data, real-time volumetric reconstruction from depth maps has become feasible and popular [2, 20, 22, 30, 33]. The data stored in the voxels is the sum of truncated signed distance functions (TSDF) to the input measurements [3]. Due to the almost outlier free nature of the input data no regularization is utilized. In order to represent large scenes, they are subdivided into multiple volumes of interest [33], or hierarchical data structures [2, 30] or voxel hashing [22] are utilized. Using stereo matching to generate depth maps from a monocular camera on a desktop GPU, [26] reconstructs small scenes interactively with high quality using a TSDF approach.

3. System Overview

On the mobile device, camera poses are computed in real-time. Our system then generates depth data and fuses it into a volume (*c.f.* Fig. 2). The camera poses are computed by a motion tracking pipeline implemented by the Tango project: KLT feature tracks [29] in the fisheye images and inertial measurements are fused in an extended Kalman filter, similar to [9]. Besides stabilizing camera tracking, IMU data is used to determine the metric scale of the scene. Our system then computes depth maps directly on the fisheye images, utilizing the wide field-of-view. These depth maps

are locally fused and filtered, which avoids integration of outlier measurements into the volume. This is crucial in outdoor scenes where often a large part of the free-space cannot be observed due to the inability to reach the required view points. Eventually, the filtered depth maps are integrated into a volume storing a TSDF using the voxel hashing [22] based approach [13]. Observations are integrated by casting rays along pixel observation directions. To benefit from the confident free-space measurements, we integrate the TSDF along each viewing ray from the camera up to the depth measurements, instead of only considering a small region around the observed depth. Meshing of the TSDF volume for display or output is done using Marching Cubes [16], which is incrementally applied to subvolumes when they change.

4. Motion Stereo on Mobile Devices

Depth maps generated from stereo matching contain significantly more outliers than data from consumer depth cameras. Integrating those depth maps directly into a volume will produce wrong geometry. The main objective of our system is to allow a user to interactively reconstruct large-scale outdoor scenes on a mobile device. In large scenes, the free-space measurements required to remove outliers often cannot be obtained. We therefore propose to identify reliable depth measurements and only use those for volumetric fusion. Consequently, we trade-off depth map completeness for accuracy. Our system displays the current state of the reconstruction to the user in real-time. Thus, the user can easily record more data where needed, making this trade-off feasible.

In order to achieve interactive frame rates, we exploit the high overlap of subsequent video frames by propagating the depth map computed from the current frame to the next using an extended Kalman filter similar to [4]. Additionally, we utilize a parallel implementation for GPUs, which nowadays are built into many mobile devices.

In the following, we describe how depth maps are obtained, propagated and filtered to remove outliers.

4.1. Depth Observation

We use plane sweep stereo [34] directly on fisheye images [8]. This allows us to make use of the whole field-of-view of the fisheye camera. We do stereo matching between the current and one recently recorded frame. This frame is selected such that it has high overlap with the current one but still provides a good triangulation angle. We do not always select the best ranked frame, but randomly select one of the three best ones to prevent using the same frame over and over again in case of little camera motion.

Cost aggregation. We use fronto-parallel plane sweeps with a fixed number of planes in a fixed depth range. The planes are sampled equidistantly in inverse depth space in analogy to disparities. We utilize a zero-mean normalized cross correlation (ZNCC) matching score in order to be robust against local and global lighting changes.

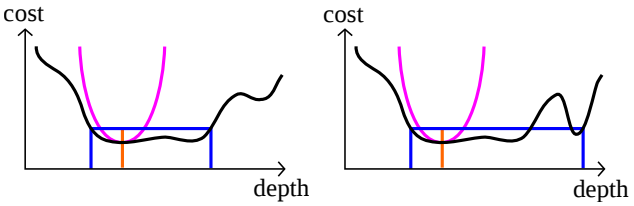

Figure 3. Depth and uncertainty extraction from the cost volume: A parabola (magenta) is fitted to the minimum of the discrete cost graph (black) for sub-plane depth estimation. The uncertainty range (blue) is estimated as the width of all minima at a factor times the minimum cost (orange). If there are multiple similar minima, the uncertainty range encompasses all of them (right).

We embed the cost aggregation in a multi-resolution scheme (using 2 levels) and compute the final cost volume as a weighted average of all cost volumes [37].

Depth extraction. After the cost volume is calculated, we extract a depth estimate and a measure of its uncertainty from it for each pixel in the image. For the depth estimate, we first find the discrete depth sample having the lowest cost for a given pixel. If the highest ZNCC score is lower than 0.4, we treat the depth observation as invalid. In order to estimate the cost minimum with sub-pixel accuracy, we use the standard approach of fitting a parabola to the three cost samples centered on the minimum (see *e.g.* [8]) and extract the depth estimate as the location of the parabola minimum (*c.f.* Fig. 3).

Uncertainty estimation. We estimate the uncertainty of each pixel's inverse depth. We are using a normal distribution as this is required to update the Kalman filter that we utilize for depth propagation. We assume that noise in the image pixel values and other un-modeled effects result in an uncertainty on the cost volume minimum. The magnitude of the noise can be modeled as $c_u = \alpha c_{\text{min}}$ for the minimum cost of a pixel c_{min} with a fixed factor α. We then find the minimum and maximum inverse depth at which the constant function with height c_u intersects the (linearly interpolated) cost curve (*c.f.* Fig. 3). Finally, the resulting uncertainty range is mapped to the symmetric uncertainty of a normal distribution. The standard deviation of this normal distribution is obtained as

$$\sigma = \max(\mu - u_{\text{min}}, u_{\text{max}} - \mu) \ .$$

This uncertainty model takes into account how well-defined the matching cost minimum in z direction is, which is affected by the geometric configuration of the cameras used for stereo and the configuration of the planes (*c.f.* Fig. 4).

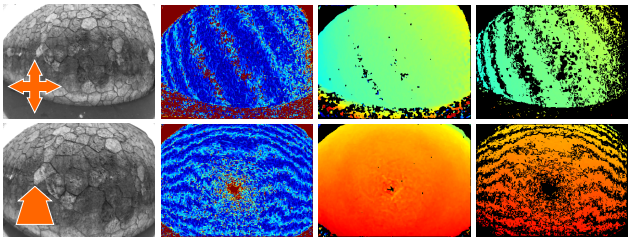

Figure 4. Uncertainty visualization for sidewards (top) and forward (bottom) movement. From left to right: camera image, uncertainty for latest plane sweep (blue: certain, red: uncertain), full internal depth map (red: close, blue: far), filtered depth map. The epipole for forward movement and parts of the wall not intersected by one of the 40 sweep planes used for this figure are uncertain.

4.2. Depth Update and Propagation

Following [4], we use an extended Kalman filter to integrate depth measurements over time and to propagate depth hypotheses from frame to frame. For each pixel in an image, we store the current state of a depth hypothesis as the inverse pixel depth μ, and the corresponding standard deviation σ. The depth observations that are obtained as described in the previous section directly provide measurements of the inverse depth μ_o with an associated variance estimate σ_o^2. If no previous estimate for a pixel exists, the observation is used to initialize the filter. Otherwise, μ_o and σ_o^2 are used to update values $\mu_{i,p}$ and $\sigma_{i,p}$ predicted from the previous iteration for iteration i as

$$\mu_i = \frac{\sigma_{i,p}^2 \mu_o + \sigma_o^2 \mu_{i,p}}{\sigma_{i,p}^2 + \sigma_o^2} \quad , \quad \sigma_i^2 = \frac{\sigma_{i,p}^2 \sigma_o^2}{\sigma_{i,p}^2 + \sigma_o^2} \quad . \tag{1}$$

After incorporating new depth observations we run a 3×3 median filter on the depth map in order to remove outliers. We evaluate the effect of this step in Sec. 5.

For correct propagation, one would need to determine the full 3D uncertainty of each pixel induced by the camera transformation. For simplicity and speed, we assume for calculation of the propagated variance that the change in depth in transforming a hypothesis from one frame to another is mostly caused by a camera translation t_z along the optical axis only, neglecting camera rotation. This assumption is justified by the expectation of relatively slow movement, in particular slow rotations, for reconstruction purposes. We map the resulting uncertainty onto the viewing ray of the corresponding pixel in the new frame. Instead of modeling the prediction noise additively on the state as in a standard extended Kalman filter, as used in [4], we model it as noise on the camera translation. This resembles the real source of error and allows to use estimates of the camera pose uncertainty for more accurate depth uncertainty propagation. Denoting the error on t_z as u_z, the state transition function returning the true state given the input becomes:

$$\mu_{i+1,\text{true}} = \left(\mu_i^{-1} - (t_z + u_z)\right)^{-1} \quad . \tag{2}$$

Figure 5. Example input images used in our system (left of each pair) and images after correcting for vignetting (right).

Consequently the propagation equations are, with $\sigma_{t_z}^2$ being the variance estimate for t_z:

$$\begin{aligned} \mu_{i+1,p} &= \left(\mu_i^{-1} - t_z\right)^{-1} \quad , \\ \sigma_{i+1,p}^2 &= J_{\mu_i} \sigma_i^2 J_{\mu_i}^T + J_{u_z} \sigma_{t_z}^2 J_{u_z}^T \\ &= \left(\frac{\mu_{i+1,p}}{\mu_i}\right)^4 \sigma_i^2 + \left(\mu_{i+1,p}\right)^4 \sigma_{t_z}^2 \quad . \end{aligned} \tag{3}$$

[4] forward-propagate the depth hypotheses from one frame to the next, resulting in holes that need to be filled separately. We convert the depth map in the old frame to a triangle mesh and render it as a dense surface to obtain propagated depth and variance values at the centers of the new frame's pixels. We impose a maximum inverse depth difference threshold for creating triangles between neighboring pixels to avoid generating triangles containing outliers or bridging depth discontinuities.

Validity counting. Similar to [4], we use a validity counter for each pixel to detect and remove inconsistent measurements. We determine the consistency of a measurement, given a prior state p and observed state o, by evaluating:

$$|\mu_p - \mu_o| < \sigma_p + \sigma_o \quad . \tag{4}$$

The validity counter is increased (up to a given maximum value) if the new measurement is consistent with the prior state. Inconsistent measurements are not integrated into the state and decrease the counter. Once the counter reaches zero, the corresponding measurement is discarded as inconsistent. Intuitively, this prevents the system from dropping a depth estimate immediately if there is a small number of outlier measurements, and prevents integrating wrong measurements into the state.

4.3. Outlier Filtering

As the volumetric TSDF representation in general is unable to handle outliers in the data, except if covered by free-space measurements, we apply a number of filtering steps to the depth maps in order to remove erroneous depth values before they are integrated into the volume.

Consistency over time. Observations which appear only for a short time are likely to be outliers or not estimated well. We make use of this for filtering them. We define a time interval $t = 0.25$ seconds for checking consistency and require to observe corresponding pixels at both points in time. Corresponding pixels are found by rendering an old depth map that was estimated approximately t seconds ago into the current view.

Variance thresholding. We use uncertainty propagation to estimate the depth variance σ_d^2 from the inverse depth variance $\sigma_{d_i}^2$ using the inversion Jacobian $J = \frac{d}{dx}\frac{1}{x}$ at inverse depth d_i:

$$\sigma_d^2 = J\sigma_{d_i}^2 J^T = \frac{1}{{d_i}^4}\sigma_{d_i}^2 \quad . \tag{5}$$

We then calculate the corresponding variance in the pixel's distance (instead of depth) and impose a threshold on it.

Angle thresholding. Common artifacts of stereo methods, *e.g.*, interpolation between foreground objects and background objects or complete outliers, produce slanted angles. Thus we set a threshold on the angle between the observation direction of each pixel and the surface normal estimated from the depth map.

Connected component analysis. This step removes remaining depth map components which are ill-defined, similar to [26]. After the angle filtering step, we transfer the sparse depth map from the GPU to the CPU and find its connected components. We remove all components which include less than 20 pixels.

5. Experimental Evaluation

Hardware and software setup. We have implemented the proposed motion stereo system both on a standard PC equipped with a Nvidia GeForce 780 GTX graphics card and Intel Core i7-4770K processor, and on a Google Project Tango Tablet Development Kit containing a Nvidia Tegra K1 (quad core) chipset. The tablet contains a fisheye camera which we use for motion stereo, as its wide field-of-view (FOV) enables a user to easily reconstruct larger parts of a scene compared to a camera with a normal FOV. Example input images from this camera can be seen in Fig. 5 on the left. In order to get more consistently colored reconstructions, we approximately correct vignetting in the images using a pre-calibrated rectification mask. Corrected images are shown on the right in Fig. 5. We downsample images to a resolution of 320×240 for all experiments. For qualitative experiments on the tablet device we use visual-inertial odometry as described in Sec. 3. For the remaining experiments we determine poses with bundle adjustment, operating on FREAK [1] descriptors for DoG [17] keypoints.

Experiments. The main objective of the proposed system is to enable an interactive reconstruction of large outdoor scenes on mobile devices. However, obtaining accurate ground truth for those scenes is a very challenging problem. Thus, we mainly perform a qualitative analysis of our system on multiple large-scale scenes that we recorded. To quantitatively measure the quality of the reconstructions obtained with our approach, we compare our motion stereo system against 3D models generated using the tablet's depth sensor on multiple indoor datasets and outdoor datasets recorded in the evening. We also evaluate against ground truth on synthetic datasets.

	Frames	Duration [s]	Length [m]
Relief	891	31.7	12.2
Exhibition	7918	299.7	176.6
Underpass	4383	166.5	70.0
Plants	2046	76.3	35.6

Table 1. Details of the datasets recorded for quantitative evaluation in this paper. Videos are recorded at roughly 27 frames per second.

	Offline	**PC**	**Mobile**
Number of sweep planes	270	200	70
Voxel resolution [cm]	2	4	7.5
Depth calculation freq. [Hz]	27	27	12
Depth integration freq. [Hz]	27	27	8

Table 2. Parameters of different configurations used for quantitative evaluation. The PC and mobile settings give approximate values for real-time use on current hardware, in medium-scale scenes.

	PC	**Tango Tablet**
Addition of a reference frame	1.5 ± 0.2	6.2 ± 3.3
Vignetting correction	1.0 ± 0.1	1.2 ± 0.7
Downsampling, transfer to GPU	0.5 ± 0.2	4.9 ± 3.0
Depth estimation	8.1 ± 0.9	83.0 ± 7.6
Plane sweep (320×240)	3.9 ± 0.6	35.2 ± 2.1
Plane sweep (160×120)	1.3 ± 0.2	13.4 ± 2.6
Multires. fusion, depth extraction	1.6 ± 0.4	15.2 ± 2.6
Depth propagation	1.3 ± 1.7	10.3 ± 5.8
Filtering, transfer to CPU	1.3 ± 0.4	13.5 ± 5.9
TSDF integration, remeshing (CPU)	30.2 ± 23.7	122.1 ± 74.6

Table 3. Timings in milliseconds (mean $\pm$ standard deviation) for individual components of our system, averaged over a sequence of multiple thousand video frames. Visual-inertial odometry, depth estimation, TSDF integration, and visualization run concurrently. The time for TSDF integration depends on the size of the scene.

5.1. Quantitative Evaluation

Synthetic datasets. We choose the living room sequences from the ICL-NUIM datasets [7], as they come with ground truth model data. The 4 short sequences show a room observed with a different camera trajectory in each sequence. The camera trajectories are real trajectories estimated by a visual odometry system. For these sequences we use a reconstruction depth range of 0.3 to 5 meters.

RGBD datasets. We compare the reconstructions obtained with our approach to 3D models generated with data by the depth sensor mounted on the Tango tablet. The depth sensor provides depth maps at around 5 Hz, which contain a very small amount of outliers. Estimating the depth of uniformly textured regions is a very challenging task for passive approaches while being trivial for RGBD sensors. We thus limit our evaluation to well-textured scenes in order to enable a fair comparison that is not biased towards the active depth sensor. Table 1 provides information about the datasets used and Figure 6 shows the reference models reconstructed with depth camera data. For these sequences

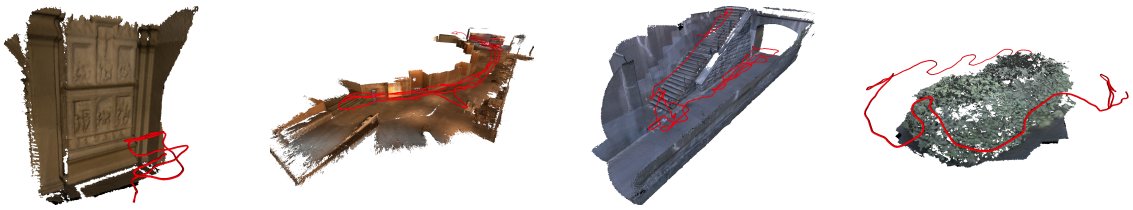

Figure 6. Models used for comparison with RGBD data, with camera trajectories in red. The meshes are generated from depth camera data of a Project Tango Tablet with the method of [13]. Datasets from left to right: *Relief*, showing a detailed wall structure within a building. *Exhibition*, a recording of a large room. *Underpass*, an urban outdoor sequence recorded in the evening. *Plants*, a natural outdoor scene consisting of two bushes.

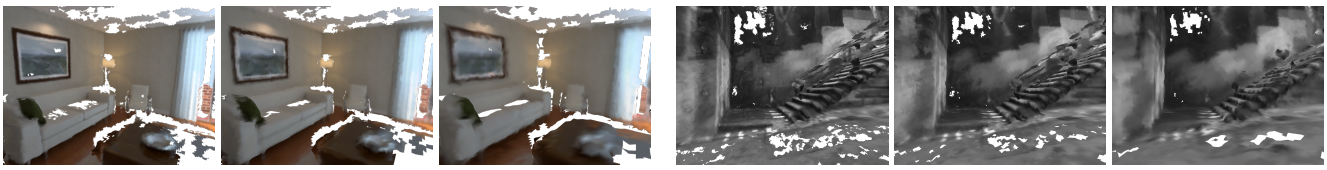

Figure 7. Example images of reconstructions with different settings. **Left:** Synthetic Living room 2 dataset. **Right:** Real-world Underpass dataset. Within each scene from left to right: Offline, PC realtime, mobile realtime.

	Depth map		Model		
	Acc.	Comp.	Acc.	Out.	Comp.
Baseline (Mobile realtime)					
Living room 0	93.2%	34.9%	89.5%	2.5%	
Underpass	89.9%	43.6%			92.9%
No connected component filtering					
Living room 0	92.4%	36.1%	86.6%	5.4%	
Underpass	89.3%	45.2%			95.5%
No angle filtering					
Living room 0	92.5%	37.2%	88.3%	3.7%	
Underpass	89.6%	44.1%			93.5%
No variance filtering					
Living room 0	93.0%	35.1%	89.2%	2.6%	
Underpass	89.8%	43.8%			93.8%
No speed filtering					
Living room 0	89.1%	52.8%	77.4%	13.1%	
Underpass	86.2%	70.7%			95.4%
No median filtering					
Living room 0	92.0%	16.4%	88.8%	1.8%	
Underpass	90.4%	27.4%			85.9%
No multiresolution					
Living room 0	94.6%	33.0%	90.4%	2.1%	
Underpass	92.0%	37.1%			88.0%

Table 4. The impact of different filtering steps on both the depth maps and resulting models, measured with mobile realtime settings by disabling the corresponding filter individually. Accuracy and completeness are evaluated at an error threshold of 10cm for the Underpass dataset with depth camera images, and 7.5cm for the synthetic Living room 0 dataset from the ICL-NUIM benchmark. Outliers are evaluated at a threshold of 15 cm.

(and all qualitative evaluations), we use a reconstruction depth range of 0.8 to 50 meters.

Results. We evaluate our system with settings suitable for interactive operation on current mobile devices, on current desktop PCs, and with settings for high-quality offline reconstruction. Table 2 gives the parameters used for the different configurations. Timings for our algorithm running on a Project Tango Tablet and on a desktop PC are given in Table 3. Components are partially run in parallel and thus do not sum up to the total frame time.

We evaluate depth map accuracy as the percentage of valid pixels in estimated depth maps which are within a certain Euclidean distance from the corresponding pixel in the ground truth depth map. Analogously, we evaluate depth map completeness as the percentage of pixels in the ground truth depth maps for which both a valid corresponding estimate exists, and its distance is within the error bound. For calculating model accuracy, we sample points on the reconstructed mesh and for each point find the distance to the closest point on the ground truth model. The percentage of points for which this distance is below a threshold is the model accuracy. We use dense Monte-Carlo subsampling to avoid introducing a bias. Analogously, for model completeness we subsample the ground truth model and find the percentage of points for which the closest point on the reconstruction is not farther away than a threshold.

As the synthetic ground truth model is always at least as complete as its reconstructions, and the reconstructions of real datasets may be at least as complete as the models generated from depth sensor data (due to the larger field of view of the fisheye camera), we evaluate only accuracy and completeness for synthetic and real datasets, respectively. For the other metric it is not clear whether missing values are due to inaccurate respectively incomplete models, or due to missing reconstruction or ground truth data. For the synthetic datasets we choose an error threshold of 7.5cm, while for the real datasets we use a threshold of 10cm to account for uncertainties in the depth sensor and the trajectory.

We evaluate the benefit of individual components of our algorithm in Table 4 (more extensive results are given in the supplementary material). Although the different filtering

	Offline				Online PC				Online mobile			
	Depth map		Model		Depth map		Model		Depth map		Model	
	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.
Living room 0	96.3%	36.2%	91.3%		96.1%	35.4%	91.1%		93.2%	34.9%	89.5%	
Living room 1	90.7%	20.5%	89.5%		90.7%	20.2%	88.3%		87.9%	24.2%	88.2%	
Living room 2	94.6%	35.1%	92.8%		94.3%	34.3%	90.8%		84.2%	26.9%	86.8%	
Living room 3	96.1%	31.6%	93.6%		95.9%	30.6%	92.1%		91.3%	30.5%	89.8%	
Relief	68.8%	32.7%		86.7%	68.7%	32.6%		84.5%	72.0%	42.8%		84.5%
Exhibition	91.7%	45.0%		94.5%	91.6%	45.1%		93.9%	90.8%	49.3%		92.9%
Underpass	90.7%	37.7%		94.3%	90.7%	37.6%		93.7%	89.9%	43.6%		92.9%
Plants	85.2%	34.3%		73.1%	85.1%	34.2%		70.2%	83.5%	37.8%		66.8%

Table 5. Results on synthetic and real sequences with different settings. Accuracy and completeness of the synthetic living room sequences are evaluated at an error threshold of 7.5cm, for the real sequences a threshold of 10cm is used.

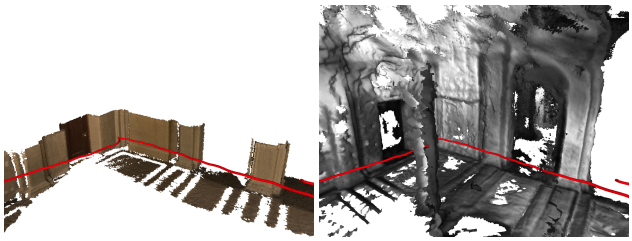

Figure 8. The completeness of the reconstruction using the fisheye camera (**right**) is significantly higher than that from the same sequence using the depth camera (**left**) due to the larger FOV and unbounded reconstruction range. Camera trajectory in red.

steps overall strongly decrease the individual depth maps' completeness, the resulting models are still very complete and accurate. Especially, they at the same time contain a low amount of outliers. For this table we evaluate model outliers analogously to model accuracy, but giving the percentage of points on the reconstruction which are farther away than 15 cm from the closest ground truth point. We argue that strong filtering is in most cases preferable to producing more complete, but possibly outlier containing models in an interactive context, as users can easily record more data if they are not satisfied with a model yet, accumulating less outliers while doing so.

Table 5 evaluates our approach on the ICL-NUIM and depth camera datasets, using all filtering steps. Depth maps in general become more accurate and complete as higher-quality settings are employed. Model completeness also rises with more processing power used. Model accuracy is very susceptible to outliers which (in contrast to static, actual geometry) accumulate in the model; this puts high-quality settings which include more frequent TSDF integration at a disadvantage. However, our offline results are in the same range as for less frequent integration. Figure 7 enables a qualitative comparison of the resulting models.

5.2. Large-Scale 3D Reconstruction

Finally, we want to demonstrate that large-scale reconstruction in outdoor environments is feasible at interactive frame rates on mobile devices. We recorded data for multiple large-scale scenes, including non-urban environments. Figure 9 shows qualitative reconstruction results obtained in real-time on a Google Project Tango tablet. Table 6 provides details on datasets we processed with offline settings; Figure 10 shows results for these datasets.

We also demonstrate the advantages of using a passive fisheye camera for 3D reconstruction. Due to the longer reconstruction range compared to depth cameras and its wide field-of-view, complete reconstructions can be created faster. Figure 8 compares reconstructions created with fisheye and depth camera data from the same camera trajectory.

6. Conclusion

We present a scalable interactive 3D reconstruction system which enables to quickly build models of indoor as well as outdoor environments. While free-space measurements provide a good means of outlier suppression for reconstructions bounded to small volumes of interest, for unbounded scenes they are often not sufficient. We therefore run several filtering steps, that we thoroughly evaluate, on the depth maps estimated by our system prior to integrating them into a global model and show that we obtain superior results.

We implemented our system for Project Tango tablets, which provides a very good motion tracking pipeline. However, we believe that the system could in general be ported to other mobile devices. One limiting factor is the computational power required by the motion stereo system. As an alternative to GPUs, FPGAs could be used for stereo computations [11]. All other components can be easily ported to other devices given proper IMU and camera calibration. Using a camera with a smaller field-of-view will negatively impact the accuracy of the tracking, but would not prevent using our approach. We assume a global shutter camera for the plane sweep stereo, however at slow movement speeds the effect of fast rolling shutters is small, and there exist methods to handle it explicitly [27].

Limitations of our system include the inherent trade-off between accuracy and completeness, for which we favor

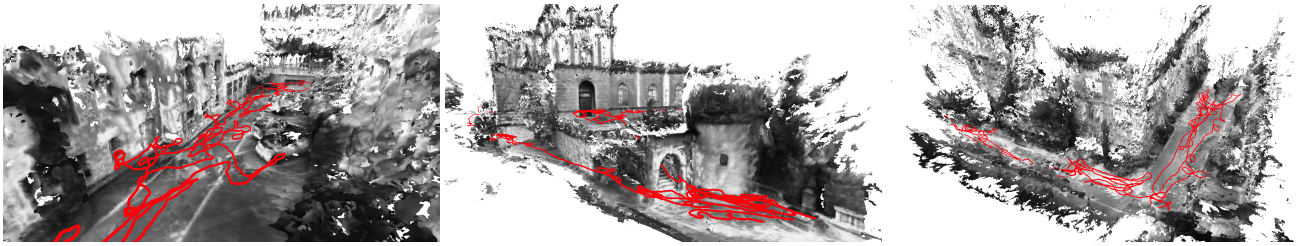

Figure 9. Examples of models generated in real-time on a Project Tango Tablet. The camera trajectories are shown in red.

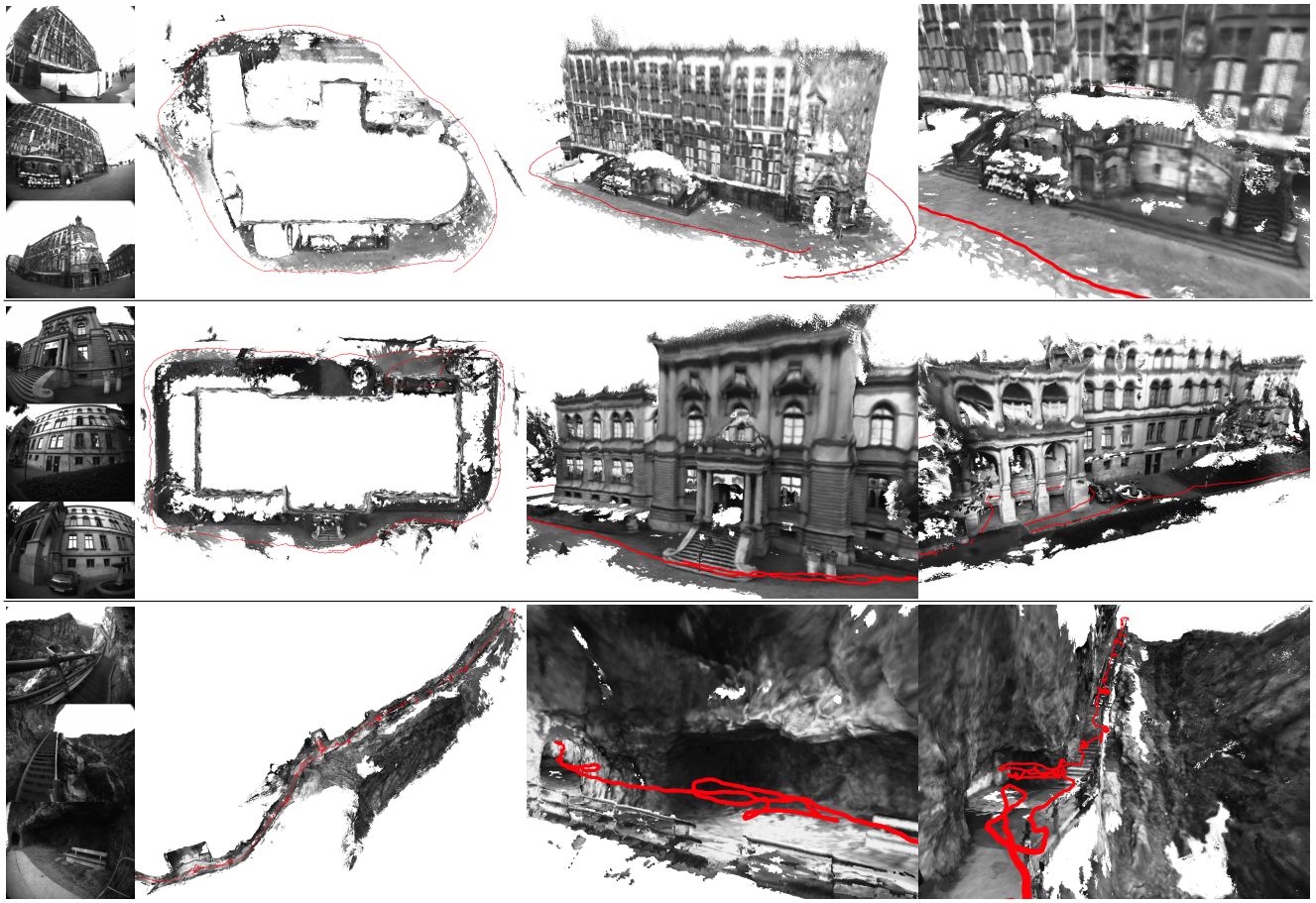

Figure 10. More demonstrations of large-scale reconstructions, processed with offline settings on bundle-adjusted trajectories; The voxel resolution was set to 4 cm to keep the mesh sizes manageable. Trajectories are shown in red. *Left:* Input image examples and orthographic projection showing the whole reconstruction from the top. *Middle and right:* Selected close-ups. Scenes, from top to bottom: Aachen Townhall, UZH Institute, Rocks at Mt. Pilatus. See the supplementary material for more images.

accuracy in an interactive setting where the user is aware of where more data is needed. Completely untextured surfaces will not be reconstructed by the system. In addition, the system does not contain means to react to on-line loop closures which change the previous trajectory. Adjusting dense volumes after loop closure is an open research topic and possible solutions were only proposed recently, *e.g.* [5].

The supplementary material to this paper, including a video, is available on the project website[1].

[1] http://cvg.ethz.ch/research/3d-modeling-on-the-go

	Frames	Duration [m:s]	Length [m]	Bundle Adj. [m:s]	Reconstr. [m:s]
Aachen Townhall	6534	3:38	226	4:58	8:06
UZH Institute	11581	7:03	281	7:14	13:29
Rocks at Mt. Pilatus	12736	7:07	236	6:36	8:33

Table 6. Details of the datasets used for Fig. 10 and timings for bundle adjustment and reconstruction on a standard desktop PC.

Acknowledgments. The research leading to these results has received funding from Google's Project Tango, from the Swiss National Science Foundation under the project Nr. 156973, and from Qualcomm.

References

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast retina keypoint. In *CVPR*, 2012. 5

[2] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. In *SIGGRAPH*, 2013. 2

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996. 2

[4] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *ICCV*, 2013. 2, 3, 4

[5] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi. Large-scale and drift-free surface reconstruction using online subvolume registration. In *CVPR*, 2015. 8

[6] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *CVPR*, 2007. 2

[7] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *ICRA*, 2014. 5

[8] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. In *3DV*, 2014. 3

[9] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. Camera-IMU-based localization: Observability analysis and consistency improvement. *The International Journal of Robotics Research*, 2013. 2

[10] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *CVPR*, 2009. 2

[11] D. Honegger, H. Oleynikova, and M. Pollefeys. Real-time and low latency embedded computer vision hardware based on a combination of FPGA and mobile CPU. In *IROS*, 2014. 7

[12] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental surface extraction from sparse structure-from-motion point clouds. In *BMVC*, 2013. 2

[13] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao. Chisel: Real time large scale 3D reconstruction onboard a mobile device. In *RSS*, 2015. 3, 6

[14] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning mobile phones into 3D scanners. In *CVPR*, 2014. 2

[15] V. Lempitsky and Y. Boykov. Global optimization for shape fitting. In *CVPR*, 2007. 2

[16] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH*, 1987. 3

[17] D. G. Lowe. Object recognition from local scale-invariant features. In *International conference on computer vision*, 1999. 5

[18] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nistér, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, 2007. 2

[19] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *CVPR*, 2010. 2

[20] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 2

[21] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011. 2

[22] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. In *SIGGRAPH Asia*, 2013. 2, 3

[23] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic feature-based on-line rapid model acquisition. In *BMVC*, 2009. 2

[24] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *ICRA*, 2014. 2

[25] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, et al. Detailed real-time urban 3D reconstruction from video. *IJCV*, 78(2-3):143–167, 2008. 2

[26] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In *ISMAR*, 2013. 2, 5

[27] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. Rolling shutter stereo. In *ICCV*, 2013. 7

[28] A. Seki, O. J. Woodford, S. Ito, B. Stenger, M. Hatakeyama, and J. Shimamura. Reconstructing Fukushima: A case study. In *3DV*, 2014. 2

[29] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994. 2

[30] F. Steinbruecker, J. Sturm, and D. Cremers. Volumetric 3D mapping in real-time on a CPU. In *ICRA*, 2014. 2

[31] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3D reconstruction on mobile phones. In *ICCV*, 2013. 2

[32] G. Vogiatzis and C. Hernández. Video-based, real-time multi-view stereo. *Image and Vision Computing*, 29(7):434–441, 2011. 2

[33] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. *Technical report*, 2012. 2

[34] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *CVPR*, 2003. 2, 3

[35] S. Yu and M. Lhuillier. Incremental reconstruction of manifold surface from sparse visual mapping. In *3DIMPVT*, 2012. 2

[36] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L1 range image integration. In *ICCV*, 2007. 2

[37] K. Zhang, Y. Fang, D. Min, L. Sun, S. Yang, S. Yan, and Q. Tian. Cross-scale cost aggregation for stereo matching. In *CVPR*, 2014. 3