2015 International Conference on 3D Vision

Super-Resolution Keyframe Fusion for 3D Modeling with High-Quality Textures

Robert Maier, Jörg Stückler, Daniel Cremers
Computer Vision Group, Technische Universität München, Germany
{maierr,stueckle,cremers}@in.tum.de

Abstract

We propose a novel fast and robust method for obtaining 3D models with high-quality appearance using commodity RGB-D sensors. Our method uses a direct keyframe-based SLAM frontend to consistently estimate the camera motion during the scan. The aligned images are fused into a volumetric truncated signed distance function representation, from which we extract a mesh. For obtaining a high-quality appearance model, we additionally deblur the low-resolution RGB-D frames using filtering techniques and fuse them into super-resolution keyframes. The meshes are textured from these sharp super-resolution keyframes employing a texture mapping approach. In experiments, we demonstrate that our method achieves superior quality in appearance compared to other state-of-the-art approaches.

1. Introduction

The wide availability of consumer RGB-D sensors has boosted research in 3D reconstruction in recent years. State-of-the-art methods in 3D model reconstruction yield impressively accurate geometric reconstruction in real-time [16, 24]. Such 3D reconstructions are well suitable for 3D printing [21]. Fast and robust estimation of high-quality visual appearance (i.e. texture) of the models has been given less attention. This plays, however, an equally important role for 3D modeling, for instance, of persons or objects.

Modern texture mapping approaches can obtain good-quality results, but are typically slow and impractical for instant 3D scanning applications. As scanning the 3D geometry with RGB-D sensors is possible in real-time, also the texture mapping process should be fast. We propose a method for fast and accurate reconstruction of geometry as well as appearance. Figure 1 shows a textured 3D model generated from low-resolution (LR) RGB-D input frames with our approach. For geometric reconstruction, we use a direct keyframe-based RGB-D SLAM method in order to estimate the camera trajectory consistently. Using these pose estimates, the individual frames are integrated into a volumetric truncated signed distance function (TSDF) representation, from which a 3D mesh is extracted. For this mesh we find a parametrization suitable for texture mapping. We significantly improve the quality of the generated texture maps through super-resolution (SR) fusion of RGB-D frames and deblurring. Simple weighted median filtering of projected color values onto the texture provides high-quality appearance results.

In experiments, we compare our method to standard pipelines that perform per-vertex coloring or texture mapping based on the original low-resolution frames. We demonstrate superior results of our method with respect to texture quality. We also evaluate the timing of our method and find that it yields high-quality results in reasonable and practical time for 3D scanning applications.

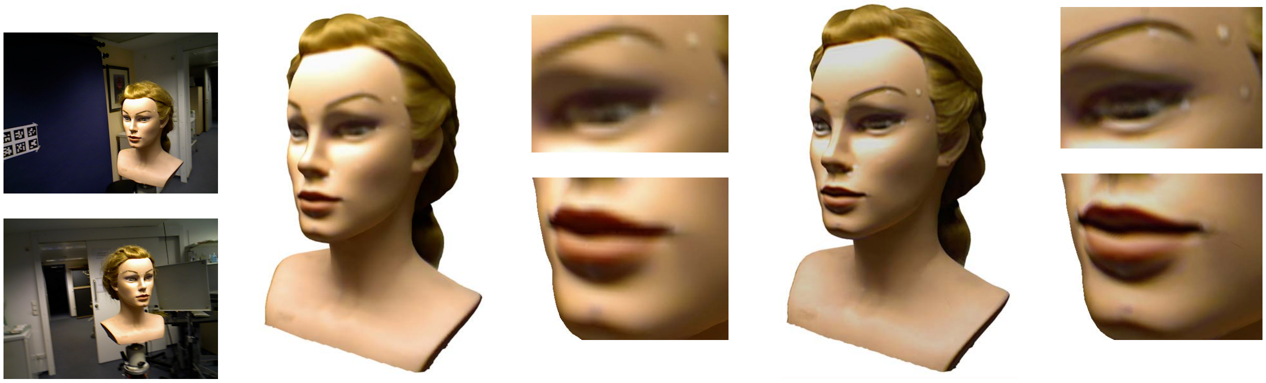
1.1. Related Work

Since the recent advent of low-cost commodity RGB-D sensors, there has been extensive research in the field of dense 3D reconstruction from RGB-D data. While generating highly accurate 3D models from RGB-D data has been investigated intensively, there seems to be a shortage of research in improving the visual appearance of such reconstructions.

To obtain geometrically accurate 3D reconstructions, Newcombe et al. [16] fuse RGB-D frames into a TSDF Volume and perform camera tracking against this model. Sturm et al. [21] developed a similar approach for reconstructing 3D printable models of persons, paired with direct TSDF tracking [1]. Other RGB-D SLAM methods [5, 20, 13, 8] are based on frame-to-(key)frame tracking with trajectory optimization and data fusion into a single model volume. Kerl et al. [9] developed a robust dense visual SLAM system that shows limited drift by combining dense robust visual odometry estimation with pose graph optimization. SLAM systems for reconstructing and mapping large-scale environments have also been developed [17, 19].

The systems presented above can produce models of

978-1-4673-8332-5/15 $31.00 © 2015 IEEE
DOI 10.1109/3DV.2015.66



(a) RGB input images (b) Vertex colors (c) Texture mapping using super-resolution keyframes

Figure 1: We propose an efficient method for generating high-quality textures from low-resolution RGB-D frames. Our approach significantly improves the visual quality of reconstructed 3D models while it is still fast enough for applicability in real-world 3D scanning scenarios.

high metric precision, however the state-of-the-art for representing the visual appearance in such 3D reconstruction systems is still volumetric averaging of per-vertex colors, such that the color resolution is limited to mesh resolution. Mostly, these vertex colors are computed as a weighted average of the observed colors for the respective vertices [16, 23, 21]. To improve the appearance, weights based on the normals computed from the depth image are employed; to remove further artifacts, pixels close to depth discontinuities are discarded.

However, to create photo-realistic 3D models of real-world objects, the challenging problem of generating and mapping high-quality textures from multiple input color images has been investigated intensively in the field of computer graphics for decades. Without increasing the geometric complexity, textures (usually of higher resolution than the mesh resolution) are mapped onto the mesh to enhance the visual quality, with camera poses assumed to be given. [15, 18] compute the texel colors using a weighted average of the observations in the input color images. Instead of using the weighted average, Coorg and Teller [2] use a color computation scheme based on weighted median to cope with color outliers in the observations. Eisemann et al. [4] correct for inaccuracies in camera poses and calibration using optical flow for mapping images to the texture map. Lempitsky and Ivanov [11] and Gal et al. [6] select a single input view per face and minimize seams, however the approaches suffer from high runtimes because of the computationally expensive combinatorial optimization. Variational super-resolution methods, e.g. by Goldlücke et al. [7], produce compelling results, paired with impractical computation times of several hours in a controlled setup with only a limited number of input views. Waechter et al. [22] texture large-scale scenes reconstructed with Structure-from-Motion, however they rely on high-quality input images and have long computation times with up to 80 minutes per dataset.

The scenario of improving the visual appearance in RGB-D based 3D reconstruction has not been tackled extensively yet. Meilland and Comport [14] fuse low-resolution images into a single high-resolution keyframe by applying a super-resolution technique. The fused keyframes exhibit an impressive level of detail, but the approach does not create a globally consistent 3D model. Recently, Zhou and Koltun [25] have shown that the colors of 3D models obtained from handheld RGB-D cameras can be improved substantially; within several minutes, they alternatingly optimize camera poses and non-rigid correction to correct for imprecise camera localization and for complex distortions resulting from inaccurate geometric models. However, they use vertex colors of an upsampled mesh, leading to an increasingly complex geometry with a still limited resolution compared to texture maps.

To the best of our knowledge, we present the first method for combining keyframe fusion with texture mapping in an RGB-D based 3D reconstruction scenario. Our practical approach is efficient, with runtimes within a few minutes, and suitable for generating high-quality texture maps from low-quality color images obtained from consumer RGB-D sensors.

1.2. Contributions

In summary, we propose a novel fast and robust 3D modeling approach that provides accurate geometry and high-quality appearance. Our method uses direct keyframe-based RGB-D SLAM to find a consistent global image alignment, and extracts a high-quality mesh from a fused TSDF representation of the images.

- The mesh is parametrized in a texture map, which we fill from fused super-resolution RGB-D keyframes.
- The super-resolution RGB-D keyframes are sharpened using image deconvolution.
- Fast texture mapping is performed using the super-resolution keyframes. High quality of the texture is obtained through weighted median filtering of the keyframe projections.

2. 3D Reconstruction System

In this section, we first describe the RGB-D sensor, the acquired data and the used camera model. We then introduce our 3D reconstruction system based on DVO-SLAM by Kerl et al. [9] and the data fusion into a TSDF volume as used by Newcombe et al. [16].

RGB-D Data Acquisition A calibrated Asus Xtion Pro Live RGB-D sensor provides us with RGB color and depth images at 30 fps at a resolution of $w \times h$ (in this case, 640×480 pixels). To limit automatic color correction during data acquisition, we fix exposure and white balance. We assume that depth and color images are registered. Since both color and depth images are utilized for real-time camera tracking, we cannot use the SXGA (1280×1024) color images provided at only 10 fps. We denote RGB images with $\mathcal{C} : \Omega_{\mathcal{C}} \to \mathbb{R}^3$ and depth images with $\mathcal{Z} : \Omega_{\mathcal{Z}} \to \mathbb{R}$.

Camera Model For the RGB-D sensor, we assume the pinhole camera model with focal length f_x, f_y and optical center c_x, c_y. The projection function π maps 3D points $\boldsymbol{p} = (X, Y, Z)^\top$ to 2D pixels $\boldsymbol{x} = (x, y)^\top$:

$$\boldsymbol{x} = \pi(\boldsymbol{p}) = \left(\frac{X}{Z} f_x + c_x, \frac{Y}{Z} f_y + c_y \right), \tag{1}$$

while 2D pixel locations $\boldsymbol{x}$ are mapped back to 3D points using their depth values $\mathcal{Z}(\boldsymbol{x})$ by the inverse projection π^{-1}:

$$\boldsymbol{p} = \pi^{-1}(\boldsymbol{x}, \mathcal{Z}(\boldsymbol{x})) = \left(\frac{x - c_x}{f_x}, \frac{y - c_y}{f_y}, 1 \right)^\top \mathcal{Z}(\boldsymbol{x}). \tag{2}$$

3D Reconstruction Framework DVO-SLAM performs dense camera tracking in real-time on the CPU and minimizes the photometric and geometric error between two RGB-D input frames to compute the relative pose. The use of color images significantly improves camera tracking and limits the drift of the SLAM system. Similarly, we perform an entropy-based loop closure detection and continuously optimize the pose graph in order to obtain a globally consistent camera trajectory.

To reconstruct a dense 3D model in a post-processing step, we fuse the N acquired RGB-D frames into a TSDF volume using their estimated absolute camera poses $\boldsymbol{T}_i = (\boldsymbol{R}, \boldsymbol{t}) \in \mathrm{SE}(3)$ (with $i \in 1 \dots N$, $\boldsymbol{t} \in \mathbb{R}^3$ and $\boldsymbol{R} \in \mathrm{SO}(3)$). We extract a 3D mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ with vertices $\mathcal{V}$ and faces $\mathcal{F}$ using the Marching Cubes algorithm. The camera poses exhibit only very limited drift due to the global pose graph optimization and hence the resulting 3D model is geometrically accurate.

3. Keyframe Fusion

Given an accurate geometric 3D model, reconstructed as described above, and the absolute camera poses for the input frames, we first fuse N_w neighboring frames into a common keyframe representation of higher resolution. We denote the color image of such a SR keyframe as $\mathcal{C}^* : \Omega_{\mathcal{C}} \to \mathbb{R}^3$ and the corresponding depth image as $\mathcal{Z}^* : \Omega_{\mathcal{Z}} \to \mathbb{R}$. To store the depth fusion weights, we introduce a depth weight image $\mathcal{W}^* : \Omega_{\mathcal{W}} \to \mathbb{R}$. These SR keyframes have the dimensions $sw \times sh$, where s is a scale factor that determines the amount of upsampling. We set the pose $\boldsymbol{T}^*$ of the SR keyframe to the first pose of the N_w LR frames to be fused. To integrate the LR images into the SR images, we additionally need to define the scale-dependent projection π_s and inverse projection π_s^{-1}, which use the upscaled intrinsic parameters sf_x, sf_y, sc_x, sc_y.

Depth Fusion We first fuse all N_w LR depth images into the corresponding SR depth image. Therefore, we compute the weights for the measured depth values, which is based on a theoretical random error model [10], as follows:

$$w_z(d) = \frac{fb}{\sigma_d} d^{-2}, \tag{3}$$

with the depth camera's focal length f, baseline b and disparity error standard deviation σ_d. Next, we transform the current depth image i into the keyframe's camera coordinate system using the relative transformation $\boldsymbol{T}^{*-1}\boldsymbol{T}_i$ between them:

$$\boldsymbol{p}^* = (X^*, Y^*, Z^*)^\top = \boldsymbol{T}^{*-1}\boldsymbol{T}_i \pi^{-1}(\boldsymbol{x}, \mathcal{Z}_i(\boldsymbol{x})). \tag{4}$$

We then use the image point $\boldsymbol{x}^* = \pi_s(\boldsymbol{p}^*)$ of the projection into the keyframe depth image to update the fused depth values and depth weights by weighted averaging:

$$\mathcal{Z}^*(\boldsymbol{x}^*) = \frac{\mathcal{W}^*(\boldsymbol{x}^*)\mathcal{Z}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x}))Z^*}{\mathcal{W}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x}))} \tag{5}$$

$$\mathcal{W}^*(\boldsymbol{x}^*) = \mathcal{W}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x})) \tag{6}$$

We achieve sub-pixel precision by updating all four neighboring depth values when transforming and projecting a depth value into the SR depth map. Occlusions are considered by fusing only the closest depth values within a given distance. After integrating all N_w LR depth images, we obtain the fused depth image $\mathcal{Z}^*$ for the SR keyframe.

Color Fusion We use the fused depth image $\mathcal{Z}^*$ to project the SR color image pixels into the LR color images. This allows us to directly look up the observed color values c_i using bilinear interpolation:

$$c_i = \mathcal{C}_i(\pi(\boldsymbol{T}_i^{-1}\boldsymbol{T}^*\pi_s^{-1}(\boldsymbol{x}, \mathcal{Z}_i(\boldsymbol{x})))). \tag{7}$$

For every observation c_i, we also compute its weight

$$w_i^c = B_i w_z(\mathcal{Z}_i(\boldsymbol{x})), \tag{8}$$

where B_i is a measure of blurriness of the color image $\mathcal{C}_i$ according to Crete et al. [3], which downweighs views with strong motion blur. Integrating the depth into the color weights enforces that objects closer to the camera obtain higher weights. We store the observed colors and weights for pixel $\boldsymbol{x}$ in its set of color observations $\mathcal{O}_{\boldsymbol{x}} = \{(c_i, w_i^c)\}$. In order to increase color fidelity, we prune observations from $\mathcal{O}_{\boldsymbol{x}}$ with missing depth values or that are within a window of 7 × 7 pixels around depth discontinuities. Instead of calculating the weighted mean for averaging the color, we calculate the weighted median, for each color channel separately:

$$\mathcal{C}^*(\boldsymbol{x}) = \arg\min_{c} \sum_{(c_i, w_i^c)\in\mathcal{O}_{\boldsymbol{x}}} w_i^c \|c - c_i\|. \tag{9}$$

Since we usually have many observations per pixel, the use of weighted median is valid, which results in an overall sharper texture. The median selects the center probable value in the distribution of colors, while the mean would be heavily affected by outliers. Integrating weights into the median allows for incorporating a confidence or a prioritization of the individual color samples.

Before fusing the LR color images into the SR keyframe, we apply a Wiener filter on these LR color images as a preprocessing step. This removes motion blur and notably improves the sharpness of the visual appearance.

Note that we perform the keyframe fusion as a postprocessing step; however, it is reasonable to perform this step online whenever a new keyframe is detected.

4. High-Quality Texture Mapping

In this section, we introduce our method for texture mapping from fused SR keyframes. First, we explain the computation of per-vertex colors based on a weighted median filtering scheme, applicable also for recomputing the vertex colors. We afterwards present our texture mapping approach, in which we compute the texel colors using the weighted median from SR keyframe color images.

4.1. Vertex Color Computation

In order to improve the colors of 3D meshes, a very common approach is to recompute the per-vertex colors of the 3D mesh vertices $v \in \mathcal{V}$. We therefore need to determine the views, in which a vertex is visible. To check if vertex $v \in \mathbb{R}^3$ is visible in view i, we render the mesh $\mathcal{M}$ into a virtual image using its pose $\boldsymbol{T}_i$ and the depth camera intrinsics. v is visible in the image, if its depth value is compatible with the depth in the depth buffer used for rendering. We then get the observed color c_i^v using bilinear interpolation:

$$c_i^v = \mathcal{C}_i(\pi(\boldsymbol{T}_i^{-1}v)). \tag{10}$$

The observation weights w_i^v of vertex v in its input views are computed as follows:

$$w_i^v = \frac{\cos(\theta)B_i}{d^2}, \tag{11}$$

where B_i is again the blurriness measure of color image $\mathcal{C}_i$ and d is the distance from v to the camera corresponding to $\mathcal{C}_i$; θ represents the angle between the vertex normal and the view vector at v for the camera. We store all color observations for vertex v and their respective weights in $\mathcal{O}_v = \{(c_i^v, w_i^v)\}$, observations close to depth discontinuities are discarded.

We can now compute the final vertex color c_v^* as the weighted mean of the observations:

$$c_v^* = \arg\min_{c_v} \sum_{(c_i^v, w_i^v)\in\mathcal{O}_v} w_i^v \|c_v - c_i^v\|^2. \tag{12}$$

Since we assume that each vertex has many observations, we can also compute the final color c_v^* (separately for each color channel) using a weighted median filtering scheme:

$$c_v^* = \arg\min_{c_v} \sum_{(c_i^v, w_i^v)\in\mathcal{O}_v} w_i^v \|c_v - c_i^v\|. \tag{13}$$

Given enough views that observe a vertex, this simple method already improves the mesh colors and results in a more detailed appearance, as demonstrated in Section 5.1.

4.2. Texture Mapping

Based on the introduced weighted median color computation scheme, we employ texture mapping to further improve the appearence of 3D models. In particular, we use the fused SR keyframes of Section 3 for texture mapping, leading to a significantly higher resolved visual appearance. We denote a texture as $\mathcal{T} : \Omega_{\mathcal{T}} \to \mathbb{R}^3$, which stores a color value at every texel $t \in \Omega_{\mathcal{T}}$.

Texture Parametrization For working with texture maps, a three-dimensional mesh needs to be projected onto a planar two-dimensional texture $\mathcal{T}$ first. We beforehand simplify the mesh geometry by decimating the number of mesh triangles. This usually results in larger triangles that can be textured more efficiently with larger patches, while the geometry is still preserved well. While different planar parametrization methods exist, our approach is in general independent of the chosen parametrization, as long as the mesh faces contain texture coordinates. In practice, we mostly use Least Squares Conformal Maps by Levy et al. [12], or a simple arrangement of the mesh triangles on the texture within a rectangular grid.

Since there is a unique mapping from a texel to its containing face, we can determine the respective surrounding vertices for each texel. The barycentric mapping $\psi : \Omega_{\mathcal{T}} \rightarrow \mathbb{R}^3$ performs a one-to-one mapping from 2D texel coordinates to 3D world coordinates. Using barycentric interpolation, we can compute interpolated 3D vertices v_t corresponding to 2D texels t and vice versa:

$$v_t = \psi(t). \tag{14}$$

Texel Color Computation To compute the texel color for every texel t in the texture map, we employ only the N^* SR keyframes $(\mathcal{C}_l^*, \mathcal{Z}_l^*)$ with camera poses $\boldsymbol{T}_l^*$ (with $l \in 1 \ldots N^*$), generated as described in Section 3.

We collect the observations of the texel by first computing its 3D vertex position v_t according to Equation (14). We then determine the set of color observations $\mathcal{O}_t = \{(c_l^t, w_l^t)\}$ for v_t analogous to Equations (10) and (11). From these observations, we compute the final texel colors by again applying a weighted median color computation scheme:

$$\mathcal{T}(t) = \arg\min_{c_t} \sum_{(c_l^t, w_l^t) \in \mathcal{O}_v} w_l^t \|c_t - c_l^t\|. \tag{15}$$

5. Experimental Results

In this section, we evaluated our approach on real-world datasets. Three evaluation sequences *face*, *phone* and *keyboard* were acquired using a handheld Asus Xtion Pro Live, details are given in Table 1. We captured RGB-D data at a low resolution of 640 × 480 pixels at 30 fps, with fixed exposure and white-balance.

The following experimental results demonstrate that (1) vertex recoloring using weighted median filtering improves the colors of 3D models compared to weighted mean, (2) fusing LR input frames into SR keyframes and using them for texture mapping improves the visual quality substantially, and (3) the proposed method is efficient and practical for real-world 3D scanning applications. All experiments were performed on a standard desktop PC with Intel Core i7-2600 CPU with 3.40GHz and 8GB RAM.

5.1. Vertex Recoloring using Weighted Median

First, we demonstrate that the visual appearance of 3D models can already be improved by using a weighted median color integration scheme. Figure 2 shows that the weighted mean in combination with discontinuity checks already improves the vertex colors significantly compared to unweighted mean. The weighted median increases the sharpness and level of detail even further and leads to a more realistic model. Still, the texture resolution is limited by the number of vertices so far. Mesh subdivision increases the number of vertices, but the increasing geometric mesh complexity makes processing the mesh intractable.

	face	*phone*	*keyboard*
# RGB-D frames	512	1359	642
# vertices (original)	159583	82942	155842
# triangles (original)	319176	165888	311686
# triangles (decimated)	40000	40000	40000

Table 1: Details of the acquired real-world datasets and the corresponding reconstructed 3D meshes.

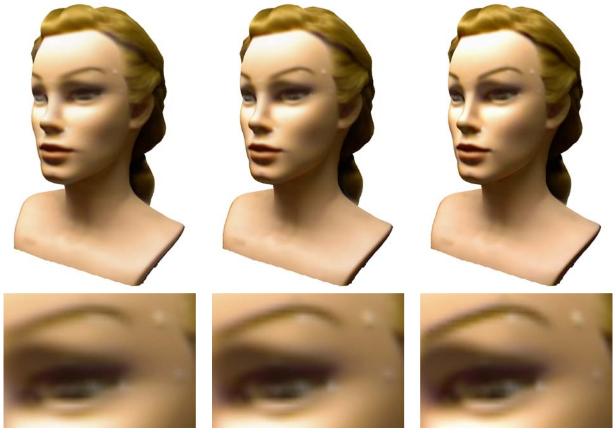
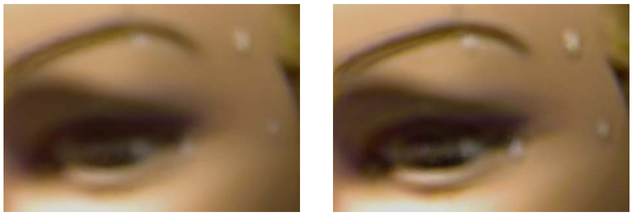

(a) Unweighted mean (b) Weighted mean (c) Weighted median

Figure 2: Improving the vertex colors of 3D models: (a) colors computed using the unweighted mean of the vertex can be improved by (b) using the weighted mean. (c) Applying the weighted median further improves the visual quality and preserves a higher level of detail.

5.2. Keyframe Fusion and Texture Mapping

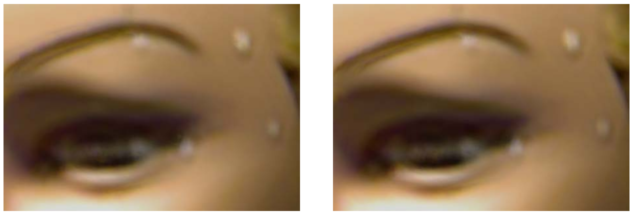
After showing that a weighted median color computation scheme has advantages compared to weighted mean, we investigate how texture mapping with weighted median filtering further improves the appearence of 3D models. In the following, we show qualitative results of texture mapping from fused SR keyframes in comparison with per-vertex colors, which serves as currently most popular state-of-the-art.

By fusing several LR color images into a SR keyframe, we obtain high-quality frames from low-quality input data.



(a) Without deconvolution (b) With deconvolution

Figure 4: (a) The textures computed from SR keyframes are substantially improved by (b) applying deconvolution (e.g. using a Wiener filter) to the input images before the keyframe fusion.



(a) Keyframes of dimensions 1280 × 960 (b) Keyframes of dimensions 2560 × 1920

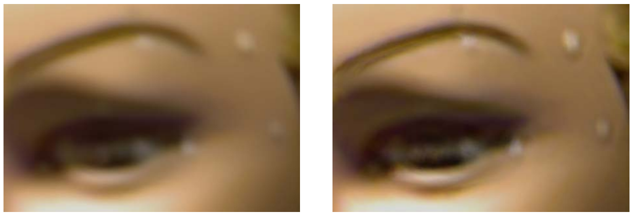
Figure 5: (a) The textures generated from keyframes of dimensions 1280 × 960 show slightly fewer details than (b) the ones generated from keyframes of dimensions 2560 × 1920.

Depending on the scale factor s, the SR images have a resolution of 1280×960 ($s = 2$) or 2560×1920 ($s = 4$). Figure 3 illustrates that both color and depth of the resulting fused SR keyframes exhibit more details compared to the LR input color images and depth maps.

An important aspect of the keyframe fusion is the deconvolution of the input images with a Wiener filter for deblurring. Figure 4 shows the results of generating a texture map for a 3D model from SR keyframes with and without deconvolution. The texture computed from the deblurred SR keyframes (Figure 4b) exhibits a sharper texture with substantially more details compared to Figure 4a. For deblurring, a Wiener filter is applied on the LR input images as a pre-processing step before fusing them into the keyframes.

Next, we compare the reconstructed surface colors depending on the scale factor s for the SR keyframe dimensions. The textures shown in Figure 5 show that the level of detail can be slightly improved by using a higher keyframe resolution of 2560×1920 ($s = 4$) compared to a resolution of 1280×960 ($s = 2$).

For comparison, Figure 6 finally shows the improvements of texture mapping with SR keyframes compared to texture mapping with the LR input images only.



(a) With LR input frames (b) With SR keyframes

Figure 6: (a) Texture mapping with LR input frames only yields inferior results compared to (b) texture mapping with SR keyframe fusion.

To demonstrate the practicability of our approach, we have reconstructed 3D models of the *face*, *phone* and *keyboard* datasets. All textures have been computed by fusion into SR keyframes of dimensions 2560 × 1920 and using weighted median filtering for computing the texel colors. As a pre-processing step, a Wiener filter has been applied to the LR input RGB images. Figures 1, 7 and 8 show the results. The texture mapped 3D models provide a photo-realistic appearance and exhibit fine surface details that are not visible in the models with per-vertex colors only.

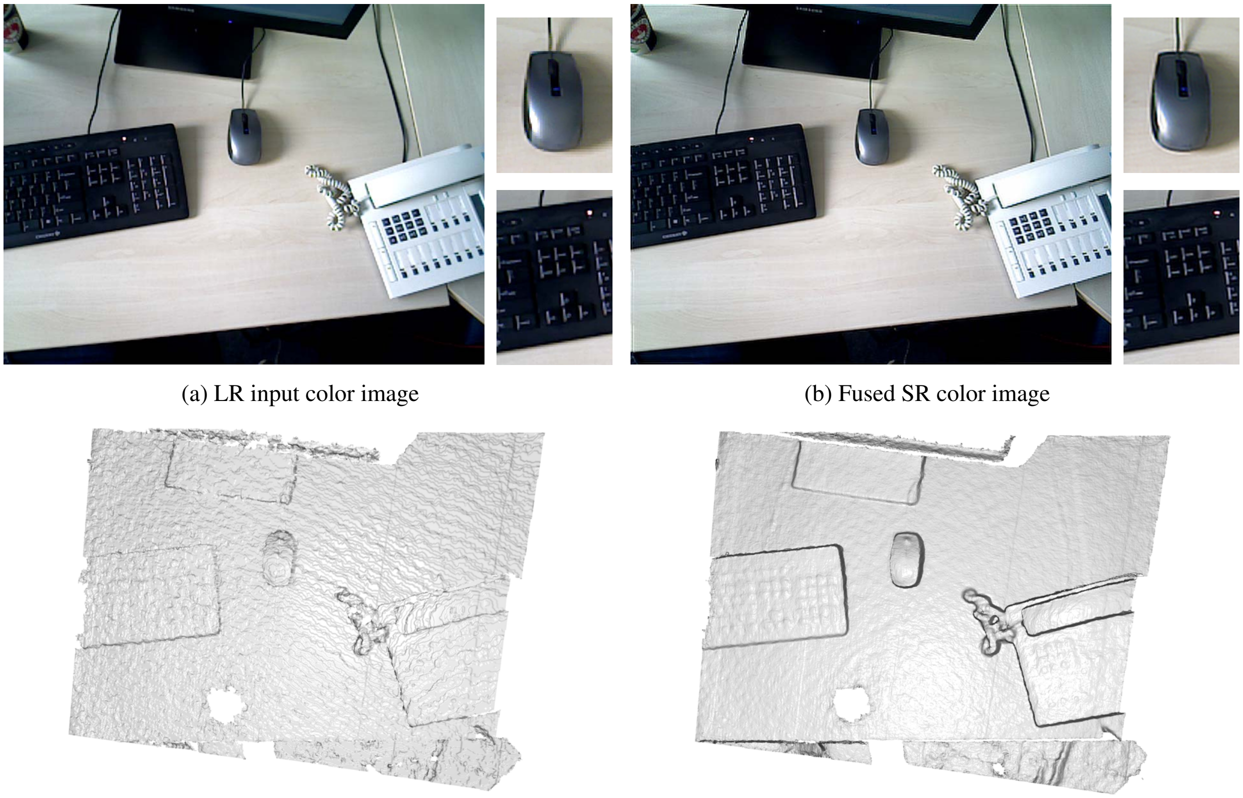
In Figure 8c, the cable at the top of the keyboard is however not represented correctly in the texture. This may either be due to inaccuracies in the estimated camera trajectory or due to an inaccurate geometric model. To compensate for this, an approach similar to Zhou and Koltun [25] must be developed, which optimizes the camera poses as well as non-rigid image corrections.

5.3. Runtime Evaluation

We finally evaluate the runtime and efficiency of the proposed texture mapping method, in particular the runtimes for keyframe fusion and texture mapping. Table 2 gives the results. With runtimes of between one and a few minutes, our approach is a very efficient method for generating high-quality texture maps. Since our implementation is based only on the CPU, a major speed-up can be achieved by porting the algorithm to the GPU. This holds in particular for the keyframe fusion, which has already been shown to work in real-time on a GPU [14].

6. Conclusion

We presented a novel efficient method for high-quality texture mapping in RGB-D-based 3D reconstruction approaches. Our method fuses low-quality color images from commodity depth sensors into super-resolution keyframes. These high-quality keyframes in turn are then mapped into a global texture for the 3D model, resulting in a significantly improved texture quality compared to simple volumetric blending. We deblur input images and use the weighted median for computing the texel colors from observations,



(a) LR input color image

(b) Fused SR color image

(c) LR input depth map (Phong shading)

(d) Fused SR depth map (Phong shading)

Figure 3: Fusing several LR input color images into a single SR keyframe allows to directly obtain high-quality color images. Compared to the LR input color image (a), the fused SR color image (b) with a resolution of 2560×1920 (scale $s = 4$) exhibits more details. Similarly, the LR input depth map (c) shows significantly more noise than the fused SR depth map (d).

		face		*phone*		*keyboard*	
	s	t [s]	fps	t [s]	fps	t [s]	fps
Texture Mapping		91.5	5.6	330.8	4.1	128.8	5.0
Keyframe Fusion	2	57.5	8.9	222.0	6.1	72.1	8.9
SR Texture Mapping	2	18.7	2.8	50.7	2.7	18.8	3.5
Keyframe Fusion	4	100.9	5.1	362.8	2.2	214.9	3.0
SR Texture Mapping	4	26.4	2.0	58.2	1.4	42.6	1.5

Table 2: Runtimes (in seconds) for texture mapping without SR keyframe fusion and texture mapping with SR keyframe fusion.

which preserves a high level of detail. Using the weighted median already provides better results for vertex coloring compared to the weighted mean. The weights in our method consider criteria such as view-angle, motion blur, and distance to the surface.

We have shown in experimental results that our method produces high-quality textures that substantially increase the photo-realism of the reconstructed 3D models. At the same time, our method is a very efficient and practical post-processing step with runtimes within a few minutes, making it useful for real-world 3D scanning application scenarios.

Acknowledgements

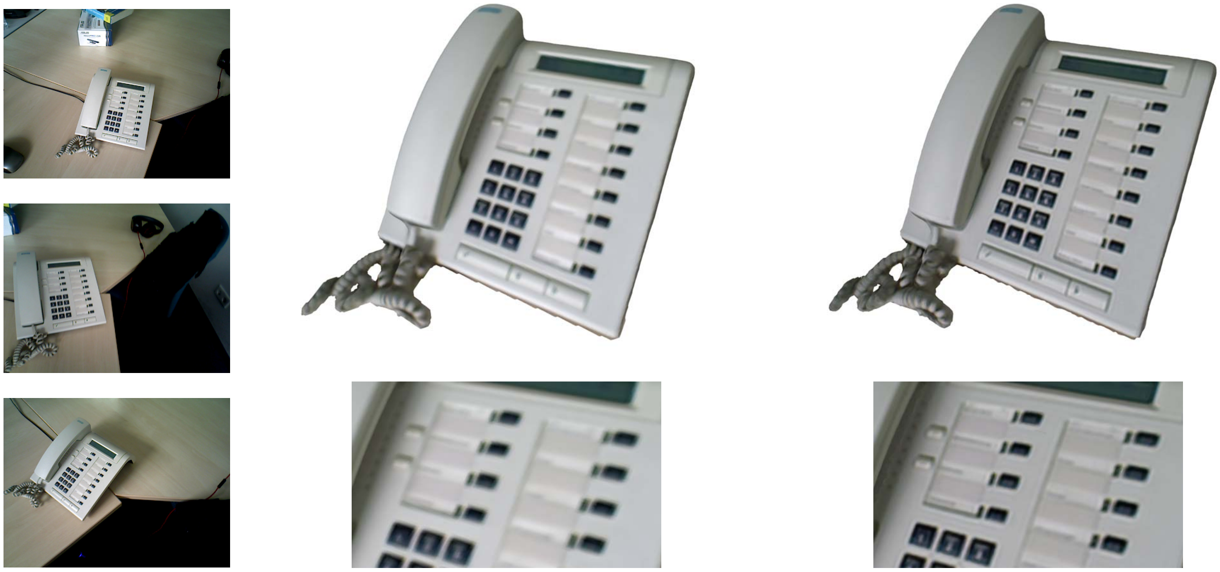
This work has been partially funded by the ERC Proof of Concept grant CopyMe3D (GA 632200) and the BMWi ZIM project 2Dzu3D (KF2080213CR4).

References

[1] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3D reconstruction using signed distance functions. In *RSS*, 2013.

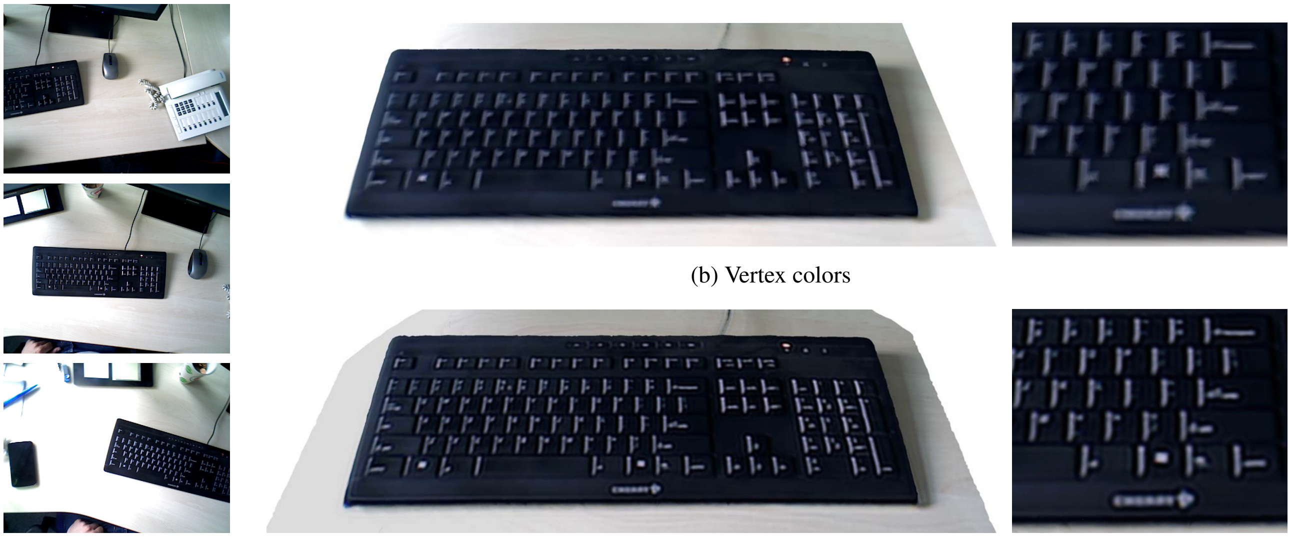
[2] S. Coorg and S. Teller. Automatic extraction of textured vertical facades from pose imagery. 1998.

[3] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *SPIE*, pages 64920I–64920I, 2007.



(a) RGB input images (b) Vertex colors (c) Texture mapping using SR keyframes

Figure 7: 3D model of the *phone* dataset reconstructed and textured using our approach: We show (a) some input images and (b) the 3D model with vertex colors only. (c) The texture mapped reconstruction provides a significantly more detailed visual appearance.



(b) Vertex colors

(a) RGB input images (c) Texture mapping using SR keyframes

Figure 8: 3D model of the *keyboard* dataset reconstructed and textured using our approach: We show (a) some input images and (b) the 3D model with vertex colors only. (c) The texture mapped reconstruction provides a significantly more detailed visual appearance.

[4] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. In *Computer Graphics Forum*, volume 27, pages 409–418, 2008.

[5] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, 2012.

[6] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or. Seamless montage for texturing models. In *Computer Graphics Forum*, volume 29, pages 479–486, 2010.

[7] B. Goldlücke, M. Aubry, K. Kolev, and D. Cremers. A super-resolution framework for high-accuracy multiview reconstruction. *IJCV*, 106(2):172–191, Jan. 2014.

[8] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *ISER*, volume 20, pages 22–25, 2010.

[9] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for RGB-D cameras. In *IROS*, 2013.

[10] K. Khoshelham and S. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[11] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *CVPR*. IEEE, 2007.

[12] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3):362–371, 2002.

[13] R. Maier, J. Sturm, and D. Cremers. Submap-based bundle adjustment for 3D reconstruction from RGB-D data. In *GCPR*, 2014.

[14] M. Meilland and A. Comport. Super-resolution 3D tracking and mapping. In *ICRA*, 2013.

[15] P. Neugebauer and K. Klein. Texturing 3d models of real world objects from multiple unregistered photographic views. In *Computer Graphics Forum*, volume 18, pages 245–256, 1999.

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

[17] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013.

[18] I. Stamos and P. Allen. 3-d model construction using range and image data. In *CVPR*, 2000.

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

[20] J. Stückler and S. Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1):137–147, 2014.

[21] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. CopyMe3D: Scanning and printing persons in 3D. In *GCPR*, 2013.

[22] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! Large-scale texturing of 3D reconstructions. In *ECCV*. 2014.

[23] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *ICRA*, 2013.

[24] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt. Real-time shading-based refinement for consumer depth cameras. *SIGGRAPH Asia*, 2014.

[25] Q.-Y. Zhou and V. Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014.