

CopyMe3D: Scanning and Printing Persons in 3D*

Jürgen Sturm¹, Erik Bylow², Fredrik Kahl², and Daniel Cremers¹

¹ Computer Vision Group, Technical University of Munich, Germany

² Center for Mathematical Sciences, Lund University, Lund, Sweden

Abstract. In this paper, we describe a novel approach to create 3D miniatures of persons using a Kinect sensor and a 3D color printer. To achieve this, we acquire color and depth images while the person is rotating on a swivel chair. We represent the model with a signed distance function which is updated and visualized as the images are captured for immediate feedback. Our approach automatically fills small holes that stem from self-occlusions. To optimize the model for 3D printing, we extract a watertight but hollow shell to minimize the production costs. In extensive experiments, we evaluate the quality of the obtained models as a function of the rotation speed, the non-rigid deformations of a person during recording, the camera pose, and the resulting self-occlusions. Finally, we present a large number of reconstructions and fabricated figures to demonstrate the validity of our approach.

1 Introduction

The advancements in 3D printing technology have led to a significant breakthrough in rapid prototyping in recent years. Modern 3D printers are able to print colored 3D models at resolutions comparable to 2D paper printers. On the one hand, the creation of a detailed, printable 3D model is a cumbersome process and represents even for a skilled graphical designer a significant amount of effort. On the other hand, the advent of consumer depth sensors such as the Microsoft Kinect has led to novel approaches to 3D camera tracking and reconstruction [5,10,19]. Probably the most prominent approach is the KinectFusion algorithm [10] that demonstrated that dense 3D reconstructions can be acquired in real-time on a GPU.

However, the resulting models are not well suited for 3D printing as they are in general incomplete when acquired with a hand-held sensor, not watertight, unsegmented, and in many other respects not optimized for cost-effective fabrication. Therefore, the combination of both technologies into a general-purpose 3D copy machine is still an open research problem. It is clear that this application bears an enormous economical potential for the future.

In our endeavor towards this goal, we investigate in this paper how an accurate 3D model of a person can be acquired using an off-the-shelf Kinect camera

* This work has partially been supported by the DFG under contract number FO 180/17-1 in the Mapping on Demand (MOD) project.

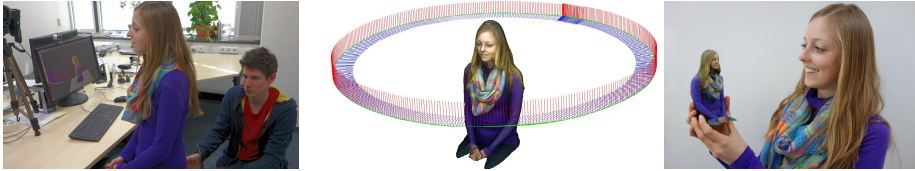


Fig. 1. We acquire the 3D model of a person sitting on a swivel chair. The acquisition process runs in real-time and displays a live view of the reconstruction model on the screen. Subsequently, the model can be printed in 3D.

and how the resulting model can be reproduced cost-effectively using a 3D color printer. Thus, our current solution corresponds to the 3D version of a classical photo booth. Our approach is based on the KinectFusion algorithm [10] for dense 3D reconstruction that we recently ameliorated for better speed and accuracy [1]. In this paper, we provide a detailed analysis of our approach, including a study of the influence of various parameters on the quality of the resulting models. Furthermore, we describe several extensions that facilitate the creation of a cost-effective, printable 3D model. Finally, we demonstrate the validity of our approach with a large number of fabricated figures.

2 Related Work

To acquire accurate 3D models of human bodies, various sensors and sensor setups have been proposed in the past [18,17]. Early 3D body scanners using multiple cameras in combination with line lasers or moiré patterns started to appear in the late 90s [7,6,20], and have many applications in medical analysis and apparel design. Camera domes consisting of up to hundreds of calibrated cameras can be used for accurate 3D reconstruction [14], but are in general not real-time capable, costly to acquire, and difficult to maintain.

Therefore, we focus in this paper on a setup that only relies on a single, hand-held sensor with which the user scans an otherwise static object. A pioneering approach was described in [13], where depth maps were computed from a combination of a structured light projector and a camera and fused into a single point cloud using ICP. More recent examples include [11,4,15] using monocular and stereo cameras, respectively. The advent of RGB-D cameras such as the Microsoft Kinect sensor further stimulated the development of approaches for live 3D reconstruction. While the first systems generated sparse representations (i.e., point clouds) of the environment [5,3], the seminal KinectFusion approach [10] used a truly dense representation of the environment based on signed distance functions [2]. Since then, several extensions of the original algorithm have appeared such as rolling reconstruction volumes [12,19] and the use of oct-trees [21]. Furthermore, the first commercial scanning solutions such as ReconstructMe, KScan, and Kinect@Home became available.

In contrast to all existing work, we propose in this paper a robust, flexible, and easy-to-use solution to acquire the 3D model of a person and a method to generate closed, watertight models suitable for 3D printing. Our work is based on our recent approach for dense reconstruction [1] that we extend here by the following components: We propose a novel weighting function to fill holes, a method for turn detection, model carving, and the automatic generation of a stand for the figure. We evaluate our approach on an extensive dataset partially acquired in a motion capture studio, and provide several examples of reconstructed and fabricated models.

3 CopyMe3D: Fast Unsupervised 3D Modeling of People

In this section, we explain how we acquire the 3D model of a person and how we prepare the model for 3D printing. Our experimental setup is as follows (see Fig. 1): The person to be scanned is sitting on a swivel chair in front of the Microsoft Kinect sensor. After the scanning software has been started, the person is rotated (by a second person) in front of the sensor. A live view of the colored 3D model is shown during scanning on the monitor to provide live feedback to the user. Scanning automatically terminates when a full turn is detected. Subsequently, a printable 3D model is generated and saved to disk. Typically, scanning takes around 10 seconds (300 frames) while the full process typically consumes less than one minute on a normal PC. In the following, we describe each of the processing steps in more detail.

3.1 Live Dense 3D Reconstruction

In this section, we briefly explain how we track the camera pose and generate the dense 3D model. We kept this section intentionally short and refer the interested reader to [10,1] for more details on signed distance functions, the KinectFusion algorithm, and our recent extensions.

Preliminaries. In each time step, we obtain a color image and a depth image from the Kinect sensor, i.e.,

$$I_{RGB} : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \text{ and } I_Z : \mathbb{R}^2 \rightarrow \mathbb{R}. \quad (1)$$

We assume that the depth image is already registered on to the color image, so that pixels between both images correspond. Furthermore, we require a signed distance function (SDF), a weight function, and a color function that are defined for each 3D point $\mathbf{p} \in \mathbb{R}^3$ within the reconstruction volume:

$$D : \mathbb{R}^3 \rightarrow \mathbb{R}, W : \mathbb{R}^3 \rightarrow \mathbb{R}, \text{ and } C : \mathbb{R}^3 \rightarrow \mathbb{R}^3. \quad (2)$$

The SDF represents the distance of each point to the closest surface, i.e., $D(\mathbf{p}) = 0$ holds for all points \mathbf{p} lying on surface, $D(\mathbf{p}) < 0$ for free space, and $D(\mathbf{p}) > 0$ for

occupied space. In the following, we treat I_{RGB} , I_Z , D , W , and C as continuous functions, but we represent them internally as discrete pixel/voxel grids (of size 640×480 and $256 \times 256 \times 256$, respectively). When access to a non-integer value is needed, we apply bi-/tri-linear interpolation between the neighboring values. We assume the pinhole camera model, which defines the relationship between a 3D point $\mathbf{p} = (x, y, z)^\top \in \mathbb{R}^3$ and a 2D pixel $\mathbf{x} = (i, j)^\top \in \mathbb{R}^2$ as follows,

$$(i, j)^\top = \pi(x, y, z) = \left(\frac{f_x x}{z} + c_x, \frac{f_y y}{z} + c_y \right)^\top. \quad (3)$$

Here, f_x, f_y, c_x, c_y refer to the focal length and the optical center of the camera, respectively. In reverse, given the depth $z = I_Z(i, j)$ of a pixel (i, j) , we can reconstruct the corresponding 3D point using

$$\rho(i, j, z) = \left(\frac{(i - c_x)z}{f_x}, \frac{(j - c_y)z}{f_y}, z \right)^\top. \quad (4)$$

In each time step, we first estimate the current camera pose ξ given the current depth image I_Z and SDF D , and subsequently integrate the new data into the voxel grids. We represent the current camera pose using twist coordinates, i.e.,

$$\xi = (\omega_1, \omega_2, \omega_3, v_1, v_2, v_3) \in \mathbb{R}^6. \quad (5)$$

These Lie algebra coordinates form a minimal representation and are well suited for numerical optimization. Twist coordinates can be easily converted to a rotation matrix $R \in \mathbb{R}^{3 \times 3}$ and translation vector $\mathbf{t} \in \mathbb{R}^3$ (and vice versa) when needed [9].

Finally, we assume that the noise of the Kinect sensor can be modeled with a Gaussian error function, i.e.,

$$p(z_{\text{obs}} | z_{\text{true}}) \propto \exp\left(-\frac{(z_{\text{obs}} - z_{\text{true}})^2}{\sigma^2}\right). \quad (6)$$

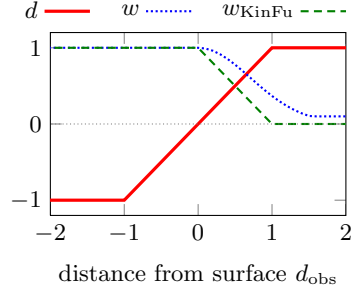
In principle, the noise of the Kinect (and any disparity-based distance sensor) is quadratically proportional to the distance, i.e., $\sigma \propto z_{\text{true}}^2$. However, as our object of interest is located at a fixed distance to the sensor (typically 1.5m) and its depth variations are relatively small, we can assume σ to be constant over all pixels.

Camera Pose Estimation. Given a new depth image I_Z and our current estimate of the SDF D , our goal is to find the camera pose ξ that best aligns the depth image with the SDF, i.e., each pixel of the depth image should (ideally) map onto the zero crossing in the signed distance function. Due to noise and other inaccuracies, the depth image will of course never perfectly match the SDF (nor will our estimate of the SDF be perfect). Therefore, we seek the camera pose that maximizes the observation likelihood of all pixels in the depth image, i.e.,

$$p(I_Z | \xi, D) \propto \prod_{i,j} \exp(-D(R\mathbf{x}_{ij} + \mathbf{t})^2 / \sigma^2), \quad (7)$$

	KinFu	Ours	Ratio
fr1/teddy	0.154m	0.089m	1.73
fr1/desk	0.057m	0.038m	1.50
fr1/desk2	0.420m	0.072m	5.83
fr1/360	0.913m	0.357m	2.56
fr1/room	0.313m	0.187m	1.67
fr1/plant	0.598m	0.050m	11.96
fr3/household	0.064m	0.039m	1.64

(a)



(b)

Fig. 2. (a) Our approach clearly outperforms KinFu on benchmark datasets in terms of the absolute trajectory error (RMSE). (b) We use a truncated signed distance function and a modified weighting function.

where $R = R(\xi)$ is a short hand for the current camera rotation, $\mathbf{t} = \mathbf{t}(\xi)$ for the camera translation, and $\mathbf{x}_{ij} = \rho(i, j, I_Z(i, j))$ for the reconstructed 3D point to keep our notation uncluttered. Note that a circular motion constraint is not imposed in the estimation process. By taking the negative logarithm, we obtain

$$L(\xi) \propto \sum_{i,j} D(R\mathbf{x}_{ij} + \mathbf{t})^2. \quad (8)$$

To find its minimum, we set the derivative to zero and apply the Gauss-Newton algorithm, i.e., we iteratively linearize $D(R\mathbf{x}_{ij} + \mathbf{t})$ with respect to the camera pose ξ at our current pose estimate and solve the linearized system.

Note that KinectFusion pursues a different (and less) effective approach to camera tracking, as it first extracts a second depth image from the SDF and then aligns the current depth image to the extracted depth image using the iteratively closest point algorithm (ICP). As this requires an intermediate data association step between both point clouds, this is computationally more involved. Furthermore, the projection of the SDF onto a depth image loses important information that cannot be used in the iterations of ICP. To evaluate the performance of both approaches, we recently compared [1] our approach with the free KinFu implementation in PCL¹ on publicly available datasets [16]. The results are presented in Fig. 2a and clearly show that our approach is significantly more accurate.

Updating the SDF. After the current camera pose has been estimated, we update the SDF D , the weights W , and the texture C similar to [2,1]. We transform the global 3D coordinates $\mathbf{p} = (x, y, z)^\top$ of the voxel cell into the local frame of the current camera $\mathbf{p}' = (x', y', z')^\top = R^\top(\mathbf{p} - \mathbf{t})$. Then we compare the depth of this voxel cell z' with the observed depth $I_Z(\pi(x', y', z'))$,

$$d_{\text{obs}} = z' - I_Z(\pi(x', y', z')). \quad (9)$$

¹ <http://svn.pointclouds.org/pcl/trunk/>

As d_{obs} is not the true distance but an approximation, d_{obs} gets increasingly inaccurate the further we are away from the surface. Furthermore, the projective distance is inaccurate when the viewing angle is far from 90° onto the surface as well as in the vicinity of object boundaries and discontinuities. Therefore, we follow the approach of [2] by truncating the distance function at a value of δ and defining a weighting function to express our confidence in this approximation:

$$d(d_{\text{obs}}) = \begin{cases} -\delta & \text{if } d_{\text{obs}} < -\delta \\ d_{\text{obs}} & \text{if } |d_{\text{obs}}| \leq \delta \\ \delta & \text{if } d_{\text{obs}} > \delta \end{cases}, \quad (10)$$

$$w(d_{\text{obs}}) = \begin{cases} 1 & \text{if } d_{\text{obs}} \leq 0 \\ \max(w_{\text{min}}, \exp(-(d_{\text{obs}}/\sigma)^2)) & \text{if } d_{\text{obs}} > 0 \end{cases}. \quad (11)$$

Note that KinectFusion uses a linear weighting function w_{KinFu} that yields a weight of zero for $d > \delta$. In contrast, our weighting function w also decreases quickly, but assigns an update weight of at least w_{min} . In this way, small holes that stem from occlusions are automatically filled. Yet, the small weight ensures that the values can be quickly overwritten when better observations become available. Experimentally, we determined $\delta = 0.02m$ and $w_{\text{min}} = 0.01$ to work well for our application. A visualization of these functions is given in Fig. 2b. We update each voxel cell with (global) 3D coordinates $(x, y, z)^\top$ according to

$$D \leftarrow (WD + wd)/(W + w), \quad (12)$$

$$C \leftarrow (WC + wc)/(W + w), \quad (13)$$

$$W \leftarrow W + w, \quad (14)$$

where $c = I_{RGB}(\pi(x', y', z'))$ is the color from the RGB image.

Both steps (the estimation of the camera pose and updating the voxel grids) can be easily parallelized using CUDA. With our current implementation, the computation time per frame is approximately 27ms on a Nvidia GeForce GTX 560 with 384 cores, and runs thus easily in real-time with 30fps.

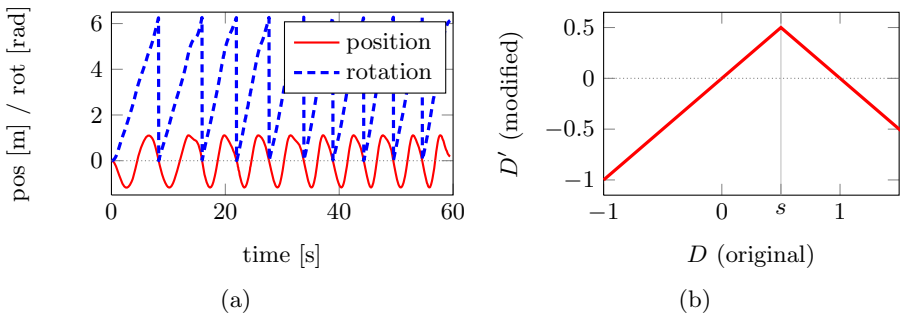


Fig. 3. (a) Analysis of our turn detection, 10 turns in a row. (b) Transfer function for the SDF to hollow out the model.

Live View and Turn Detection. With the algorithm described above, we obtain an estimate of the signed distance and color for every cell of the voxel grid. To display this model to the user, we copy the data every two seconds from the GPU to the CPU (which consumes 60ms) and run a threaded CPU implementation of the marching cubes algorithm [8]. The mesh generation takes around between 1000 and 1500ms on a single CPU core. The resulting triangle mesh typically consists of approximately 200.000–500.000 vertices (and faces), that we display together with the estimated camera trajectory to the user using OpenGL (see Fig. 1).

We implemented a simple strategy to detect when a full 360° turn is complete. To achieve this, we summarize the angular motion of the camera, i.e., we compute the motion between two consecutive frames and determine the rotation angle according to [9], i.e.,

$$\alpha_t = \cos^{-1}(\text{trace}(R_{t-1}^\top R_t) - 1). \quad (15)$$

We terminate data acquisition when $\sum_t \alpha_t > 2\pi$. Figure 3a compares the estimated rotation angle in comparison to the (absolute) position of the chair. We use this principle to automatically stop data acquisition after the person has performed a full turn.

3.2 Model Post-processing

While live feedback is important for the user (e.g., to find the right position for the chair), we found that further post-processing is required for 3D printing.

The first problem is that the triangle mesh is unclosed where objects stick out of the reconstruction volume. This is in particular the case at the bottom of the body, for which no triangles are generated. To remedy this problem, we augment the SDF by an additional layer from all sides with $D = -\delta$, which ensures that the triangle mesh is guaranteed to be closed and watertight and thus 3D printable (see Fig. 4a for a profile).

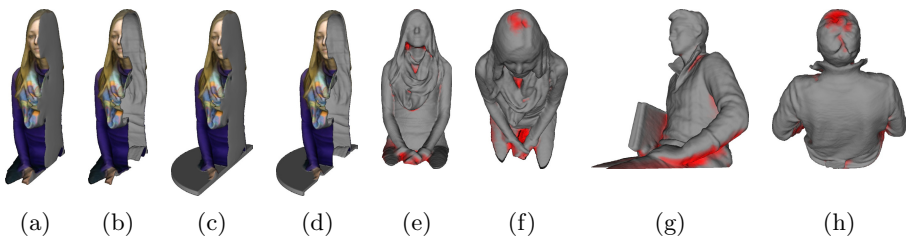


Fig. 4. Illustration of various parameters of the model acquisition process. (a) Profile of the solid model, (b) hollow model ($s=0.01\text{m}$), (c) automatically added solid stand, (d) hollow version. (e–g) Self-occlusions for a camera at eye height based on the accumulated weights W .

Depending on the body posture, the center of mass will not be above the footprint of the object, and thus the fabricated figure would fall over. To prevent this, we automatically add a small cylindrical stand to the model as shown in Fig. 4c+d. To this end, we first determine the center of mass x_m, y_m (by summing the positions of all voxel cells with $D > 0$) and the radius r of the model. Subsequently, we add a cylindrical disk below the model by updating each cell (x, y, z) of the SDF with $z < z_{base}$ according to

$$D = r - \sqrt{(x - x_m)^2 + (y - y_m)^2}. \quad (16)$$

Finally, we found that the printing costs prevalingly depend on the amount of the printed material, i.e., the printing costs can be reduced by a factor of five when the model is hollow. To achieve this, we transform the values in the SDF according to

$$D' = \begin{cases} s - D & \text{if } D > s/2 \\ D & \text{otherwise} \end{cases}, \quad (17)$$


so that voxels that are located deeper than s within the object will be mapped to the outside afterwards. This transfer function is visualized in Fig. 3b. However, our estimate of the SDF is only a coarse approximation, so that the above transformation leads to strong artifacts and unconnected components. Therefore, we re-compute the SDF with the correct signed distance values by finding the nearest neighbor from the pre-computed triangulated mesh. To speed up this computation, we insert all vertices into an oct-tree structure and only update voxel cells within the object (i.e., cells with $D > 0$). Example profiles of the resulting models are given in Fig. 4b+d.



Fig. 5. More examples of 3D models acquired with our approach and the resulting printed figures

4 Results

Figures 1 and 5 show the acquired 3D models and printed figures of several persons. To obtain a high quality model, we observed that a raw beginner needs around two to three trials to reduce body motions and articulations during model acquisition. For example, a small head motion during recording can already lead to significant inaccuracies in the reconstructed model, such as a double nose.

	condition	rot. time [s]	deformation [mm]
	external rotation	7.0 (\pm 0.8)	22.3 (\pm 21.7)
	external, fast	2.7 (\pm 0.3)	15.8 (\pm 4.8)
	self rotation	6.2 (\pm 0.4)	46.2 (\pm 15.3)
	self, fast	2.6 (\pm 0.8)	60.8 (\pm 28.2)
	mannequin	7.8 (\pm 1.7)	8.9 (\pm 4.8)
	mannequin, fast	4.8 (\pm 1.7)	11.4 (\pm 1.3)

(a)

(b)

Fig. 6. Evaluation of non-rigidity due to body motions during model acquisition in a motion capture studio (see text). (a) Experimental setup. (b) Results.

To better understand the source and magnitude of body motions during recording, we tracked the positions of 15 distinct visual markers in a motion capture studio (see Fig. 6a). We tested four different conditions, i.e., external vs. self-propelled rotation, slow vs. fast rotation speed, and real persons vs. a shop-window mannequin. We averaged the results over five different persons with ten trials each. The results are given in Fig. 6b. We obtained the best results (least body motions) when the swivel chair was rotated quickly (3s) by a second person.

Furthermore, we analyzed the amount and locations of self-occlusions as a function of the camera configuration. We tried three different camera heights (at chest height, at eye height, above head) and inspected the resulting models. While all camera positions inevitably lead to some self-occlusions, we found that positioning the camera at eye height leads to the visually most pleasing result (see Fig. 4e–h for a visualization of the resulting self-occlusions).

Lastly, we evaluated how far we can lower the frame rate of the Kinect sensor before our reconstruction algorithm fails. Here, we found that our algorithm typically diverges for frame rates below 6 Hz for the slow and 15 Hz for the fast rotation speed. It should be noted that our algorithm never failed over ten trials on all subjects when operated at the full frame rate (30 Hz), neither for the slow, nor for the fast rotation speed. Therefore, we conclude that our approach is highly robust and still bears significant potential for further reduction of the computational requirements.

5 Conclusion

In this paper, we presented a novel approach to scan a person in 3D and reproduce the model using a color 3D printer. Our contributions on the scanning side include an efficient solution to camera tracking on SDFs, an improved weighting function that automatically closes holes, and a method for 3D texture estimation. To prepare the model for 3D printing, we described a technique to generate a closed, watertight model, to automatically add a stand, and to make it hollow. We evaluated the robustness of our algorithm with respect to the frame rate and rotation speed, and the severity of self-occlusions as a function of the camera pose. We presented a large number of 3D models from different persons and

the corresponding printed figures. With this work, we hope to contribute to the development of a general, low-cost 3D copy machine.

References

1. Bylow, E., Sturm, J., Kerl, C., Kahl, F., Cremers, D.: Real-time camera tracking and 3D reconstruction using signed distance functions. In: RSS (2013)
2. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: SIGGRAPH (1996)
3. Engelhard, N., Endres, F., Hess, J., Sturm, J., Burgard, W.: Real-time 3D visual SLAM with a hand-held camera. In: RGB-D Workshop at ERF (2011)
4. Fuhrmann, S., Goesele, M.: Fusion of depth maps with multiple scales. *ACM Trans. Graph.* 30(6), 148 (2011)
5. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In: ISER (2010)
6. Horiguchi, C.: BL (body line) scanner: The development of a new 3D measurement and reconstruction system. *Photogrammetry and Remote Sensing* 32 (1998)
7. Jones, R., Brooke-Wavell, K., West, G.: Format for human body modelling from 3D body scanning. *International Journal on Clothing Science Technology* (1995)
8. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics* 21(4), 163–169 (1987)
9. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.: *An Invitation to 3D Vision: From Images to Geometric Models*. Springer (2003)
10. Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: ISMAR (2011)
11. Newcombe, R., Lovegrove, S., Davison, A.: DTAM: Dense tracking and mapping in real-time. In: ICCV (2011)
12. Roth, H., Vona, M.: Moving volume KinectFusion. In: BMVC (2012)
13. Rusinkiewicz, S., Hall-Holt, O., Levoy, M.: Real-time 3D model acquisition. In: SIGGRAPH (2002)
14. Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: CVPR (2006)
15. Strobl, K., Mair, E., Bodenmüller, T., Kielhöfer, S., Sepp, W., Suppa, M., Burschka, D., Hirzinger, G.: The self-referenced DLR 3D-modeler. In: IROS (2009)
16. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IROS (2012)
17. Weiss, A., Hirshberg, D., Black, M.: Home 3D body scans from noisy image and range data. In: ICCV (2011)
18. Werghi, N.: Segmentation and modelling of full human body shape from 3D scan data: A survey. In: VISAPP (2006)
19. Whelan, T., McDonald, J., Johannsson, H., Kaess, M., Leonard, J.: Robust tracking for dense RGB-D mapping with Kintinuous. In: ICRA (2013)
20. Winkelbach, S., Molkenstruck, S., Wahl, F.M.: Low-cost laser range scanner and fast surface registration approach. In: Franke, K., Müller, K.-R., Nickolay, B., Schäfer, R. (eds.) DAGM 2006. LNCS, vol. 4174, pp. 718–728. Springer, Heidelberg (2006)
21. Zeng, M., Zhao, F., Zheng, J., Liu, X.: Octree-based fusion for realtime 3D reconstruction. *Graphical Models* 75 (2012)