Robust Camera Tracking by Combining Color and Depth Measurements

Erik Bylow
Lund University
Mathematical Sciences

Carl Olsson
Lund University
Mathematical Sciences

Fredrik Kahl
Lund University
Mathematical Sciences

Email: {erikb,calle,fredrik}@maths.lth.se

***Abstract*—One of the major research areas in computer vision is scene reconstruction from image streams. The advent of RGB-D cameras, such as the Microsoft Kinect, has lead to new possibilities for performing accurate and dense 3D reconstruction. There are already well-working algorithms to acquire 3D models from depth sensors, both for large and small scale scenes. However, these methods often break down when the scene geometry is not so informative, for example, in the case of planar surfaces. Similarly, standard image-based methods fail for texture-less scenes. We combine both color and depth measurements from an RGB-D sensor to simultaneously reconstruct both the camera motion and the scene geometry in a robust manner. Experiments on real data show that we can accurately reconstruct large-scale 3D scenes despite many planar surfaces.** [1]

I. Introduction

The capability to acquire 3D models has been one of the major challenges in computer vision for a long time and it is still an active research area. Using a 3D depth sensor together with a GPU, this can even be done in real-tme. These 3D models have applications in several areas: robotics, augmented reality, medical imaging are a few of them. In robotics, for instance, it is necessary to know the position of the camera to perform navigation control, and by using the 3D model one can also do object recognition and path planning. In other areas, such as refurbishment, one can plan the work better by doing measurements in the models before the work actually starts.

In computer vision, the process of determining camera positions and scene structure is known as the structure from motion problem. Typically one extracts sparse feature points from a set of images using a keypoint detector such as SIFT, [5], then these points are matched between the images. Camera pose and 3D point positions that minimize the reprojection error are then computed using bundle adjustment techniques. In a subsequent step, a dense surface representation can be estimated, see for example, [8] when the cameras are known. For smaller scenes, it has been shown that dense 3D reconstruction in real-time is possible by using an ordinary hand held camera [11, 6].

However, using cheap depth sensors such as the Microsoft Kinect and the Asus Pro Live sensor opens up new possibilities for doing 3D reconstruction. The most well-known approach is perhaps KinectFusion Newcombe et al. [7]. This work showed that it is possible to obtain a dense 3D model of a medium sized room in real-time. The geometry is represented using a Signed Distance Function (SDF) and Iterated Closest Point (ICP) method is used to estimate the camera motion. This approach works well for small and medium sized rooms but it is memory consuming. In Whelan et al. [14], the approach was extended by using a rolling volume to do large scale reconstruction. However, the tracking they use is based on Steinbruecker et al. [9] together with the ICP-based Kinect-Fusion approach. A disadvantage is that the former is prone to drift and it is not possible to recover a reconstructed scene if you return to a previous spot (that is, no automatic loop-closure). Lately, Steinbruecker et al. [10] resolved the problem of recovering the scene when returning to a previous location by using an octree structure to represent the geometry. The results are clearly impressive and the tracking is accurate, but they need both loop closure techniques and graph optimization to get a globally accurate camera trajectory.

In Bylow et al. [1], it was shown how the signed distance function could be directly used to estimate the camera trajectory for simultaneous real-time 3D reconstruction and camera tracking. It was experimentally demonstrated that this approach outperforms the ICP-based KinFu implementation, which is an open source implementation of KinectFusion. A disadvantage is that the signed distance function is represented as a uniform grid which requires a lot of memory. It is purely based on depth measurements, which makes it sensitive for scenes with little geometric scene structure.

The main contribution of this paper is that we show how the camera trajectory and the 3D scene geometry can be robustly estimated by extending the tracking algorithm from Bylow et al. [1] by combining color and depth measurements. That is, we used the colourized 3D model represented as a textured signed distance function to do global frame-to-model tracking directly. This has the advantage, compared to ICP, that we can extract information for all 3D points and use it for tracking, where ICP needs to do data association and find corresponding point-pairs which gives less points to extract information from. Additionally, we also show how a more memory efficient representation of the signed distance function makes it possible to perform larger scale reconstructions. In contrast to most other well-known tracking methods, which are sensitive to either lack of geometric structure (for instance, ICP-methods like KinectFusion), or lack of texture and visual features (standard feature based methods), we show how this extension can handle both scenes with no structure as well as scenes with no texture.

[1] We gratefully acknowledge funding from the Swedish Foundation for Strategic Research (Future Research Leaders), the Swedish Research Council (grants no. 2012-4213 and 2012-4215) and the Crafoord Foundation.

II. Related Work

KinectFusion [7] was one of the first methods to show the potential of consumer depth sensors. Similar to our approach, they use a truncated signed distance function to represent the surface geometry in a uniform voxel grid. However, to track the camera they use the standard ICP method, and they do not use any color information in the tracking. Recently, Chen et al. [2] showed how KinectFusion can be extended for larger scenes through a more memory efficient representation, but the tracking is still based on the standard ICP method which is sensitive to scenes with little scene structure.

There are several previous attempts that include photometric information in the tracking algorithm. Most algorithms are either feature based like Endres et al. [4] or based on photo consistency as in Steinbruecker et al. [9]. For example, Whelan et al. [14] uses Steinbruecker et al. [9] together with the KinectFusion approach to track the camera in large scale scenes in order to make it robust for scenes with either little texture or little structure. However, the use the model to track with respect to structure and frame-to-frame for tracking with respect to photo consistency. Even though they estimate a colourized 3D model, they do not use any color information from the model in the tracking. Moreover, by extracting the distance and color information directly, rather than doing ray tracing to extract another depth image as in KinectFusion, we gain more information since we do not have to do any data association to find corresponding point-pairs, which was already shown in [1] for structure based tracking. To efficiently represent the geometry, they use a rolling volume. This has the drawback that when leaving a part of the room, and that part is saved to the hard drive, it is not possible recover it when returning to the same place later.

Very recently, Steinbruecker et al. [10] showed that octrees can be used to represent the 3D model to achieve a memory efficient and global representation. The tracking approach is based on Steinbruecker et al. [9] but they extend it by including the depth images in their framework, i.e. geometry is also taken into account. However, they do tracking between current frame and a key-frame to reduce drift together with global graph optimization and loop closure to get a globally consistent trajectory. Without loop closure and the graph optimization, the drift would be significant over a large scale. In contrast we show, how structure information and color information can be combined by using the global 3D model directly.

The same holds for Endres et al. [4] which finds corresponding points between consecutive images and performs ICP between sparse 3D points. To reduce drift, global graph optimization is carried out and the geometry is represented using an occupancy grid in an octree. Unlike our method, Endres et al. [4] will fail in scenes with structure but without visual features.

In [13], a combination of ICP and photometric error minimization is applied where corresponding points in two consecutive images are detected. The methods then minimizes both the depth distance and the photometric error between these corresponding points. This is done in a frame-to-frame manner and only the tracking problem is addressed.

In contrast to all previous work, we show how a textured surface represented as a signed distance function be can used to do global tracking for large-scale scenes and we show that it is robust in scenes where there is either no structure or no visual feature information. We also do a quantitative evaluation of how the color integration affects the tracking using different datasets from [12].

III. Theory

Here we present our approach to represent the scene geometry and how the camera tracking problem is solved. It works by iteratively updating the pose of the camera and the scene geometry for every new RGB-D image acquired.

A. Notation and Preliminaries

In this paper we use the pinhole camera model. We assume that the focal lengths f_x and f_y and principal point (c_x, c_y) are known. Given a pixel coordinate (u, v) and a corresponding depth $z = z(u, v)$, its 3D point $\mathbf{x} \in \mathbb{R}^3$ can be computed by

$$\rho(u,v) = \left(\frac{(u - c_x)z}{f_x}, \frac{(v - c_y)z}{f_y}, z \right). \tag{1}$$

Conversely, given a 3D point $\mathbf{x} = (x, y, z)$ its projection can be computed from

$$\pi(\mathbf{x}) = \left(\frac{x f_x}{z} + c_x, \frac{y f_y}{z} + c_y \right). \tag{2}$$

At each time step t we receive a color image $I_c^t \subset \mathbb{R}^2$ and a depth image $I_d^t \subset \mathbb{R}^2$. We model these using the functions

$$I_c^t \to [0,1]^3 \text{ and } I_d^t \to \mathbb{R}. \tag{3}$$

Furthermore, the rotation and translation of the camera is denoted by $R \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$.

B. Pose Estimation

In this section present a method for computing the pose of a new camera given the current estimate of the scene surfaces. The surface geometry is represented in a voxel grid $\Omega \subset \mathbb{R}^3$ using a (truncated) signed distance function (SDF)

$$\psi : \Omega \to \mathbb{R}. \tag{4}$$

The SDF gives for an arbitrary point $\mathbf{x} \in \Omega$ the signed distance to surface and the actual surface is obtained by computing the zero level-set. To represent the texture of the surface we use the function

$$\psi_{RGB} : \Omega \to \mathbb{R}^3, \tag{5}$$

which gives the RGB intensities at position $\mathbf{x} \in \Omega$.

Given a new pair of color and depth images I_c^t and I_d^t, the goal is to find the pose of the new camera. To do this we attempt to find a global rotation R and translation $\mathbf{t}$ of the camera such that

(i) the difference between the current surface estimate and the surface obtained from the depth image I_d^t is minimized (see Figure 1) and at the same time,

(ii) the difference between the estimated surface texture and the color image I_c^t is minimized (see Figure 2).

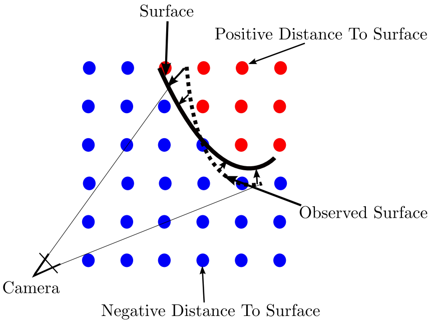


Fig. 1. The goal is to find a global rotation and translation of the camera so that the observed surface is consistent with the current depth. The dotted line is the the surface seen from the camera and that should fit as good as possible to the filled line which is the estimated surface.

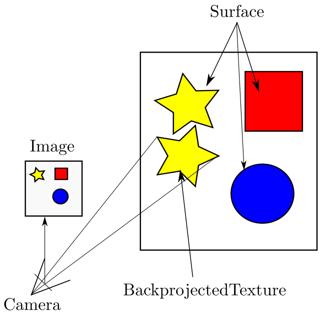


Fig. 2. To the right we have the textured surface, by using the depth information we can reconstruct the seen surface and see how well the backprojected texture fits with the observed texture on the surface. In this example the backprojected star should coincide with the star on the surface if the rotation and translation was right.

Since we use both geometric and photometric information, this results in an objective function that is robust for situations when the geometric scene structure may be ambigous or when the RGB image is not informative enough.

We define the photometric error for a voxel $\mathbf{x}$ as

$$\phi(\mathbf{x}) = \|\Sigma(\psi_{RGB}(\mathbf{x}) - I_c^t(\pi(\mathbf{x})))\|. \tag{6}$$

Like [14] we use the grayscale and accordingly weight the errors in the different channels with

$$\Sigma = \begin{pmatrix} \sqrt{0.299} & 0 & 0 \\ 0 & \sqrt{0.587} & 0 \\ 0 & 0 & \sqrt{0.114} \end{pmatrix}. \tag{7}$$

Assuming that the photometric and geometric errors are independent, it is natural to define the total error as

$$E(R,\mathbf{t}) = \sum_{i,j} \left(\psi(R\mathbf{x}_{ij} + \mathbf{t})^2 + \alpha\phi(R\mathbf{x}_{ij} + \mathbf{t})^2\right), \tag{8}$$

where α is a positive weighting factor. Here the sum is taken over all pixels (i,j) in the image, which have corresponding 3D points $\mathbf{x}_{ij} = \rho(i,j)$, cf. (1).

To minimize this objective function, we switch notation and use the Lie-algebra representation of the camera pose instead. In the Lie-algebra representation, a rigid body motion is represented as a 6-dimensional vector

$$\boldsymbol{\xi} = (r_x, r_y, r_z, t_x, t_y, t_z). \tag{9}$$

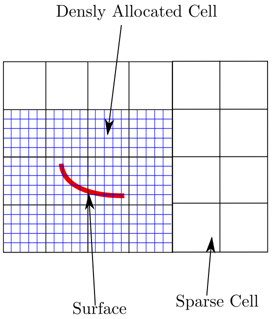


Fig. 3. To use the memory more efficient, we allocate only a small dense grid close to the surface, as illustrated in the figure.

With this representation, the residual vector can be written

$$r_{ij}(\boldsymbol{\xi}) = \left[\psi(R\mathbf{x}_{ij} + \mathbf{t}), \sqrt{\alpha}\phi(R\mathbf{x}_{ij} + \mathbf{t})\right]^T, \tag{10}$$

and we can rewrite the error function as

$$E(\boldsymbol{\xi}) = \sum_{i,j} r_{ij}(\boldsymbol{\xi})^T r_{ij}(\boldsymbol{\xi}). \tag{11}$$

which we minimize using the standard Gauss-Newton method.

C. Representation of the Geometry and Estimation of the SDF and Colors

In this section we show how we represent the signed distance function as a sparse voxel grid and how we estimate the colors and the signed distance for each voxel.

1) Representation of the voxel grid: The pose estimation relies on that we already have an estimation of ψ and ψ_{RGB}, i.e. we have an estimation of the already seen textured surface. Thus we need a method of estimating the texture and distance and a way of integrating these measurements into the grid.

Our approach relies heavily on the the work by Curless and Levoy [3], where a simple and fast algorithm to fuse several depth images into a distance function is presented. They do however use a uniform voxel grid. This requires a lot of memory, and most of this memory is used to represent free space. We try to overcome this by using a sparse representation instead of a uniform grid. Recently, [2] presented a similar way of representing the voxel grid in a more memory efficient way.

The basic idea is that the SDF only needs to be densly sampled in the vicinity of the surface. We therefore allocate a very sparse voxel grid, which can cover a large volume without requiring to much memory. Then we can detect in which one of these sparse cells the surface is present and make a denser allocation in that particular cell and its neighbours. The goal is to not waste memory to represent free space, but only have dense representation close to the surface, which is illustrated in Figure 3.

As we get more information and see new parts of the scene, the grid is recomputed for each new image, i.e. if we detect surface in a cell that has not densely allocated before, we do it now.

The reason to why we need to also allocate the neighbours to the cell in which the surface is detected is that our tracking relies on that we can extract information in the voxel grid. If

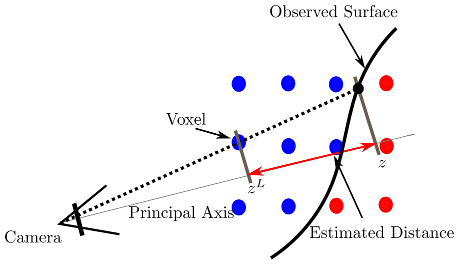


Fig. 4. The idea behind the projected point-to-point metric is to measure the projected distance between the observed surface and the voxel.

only the cell which the surface is detected in was allocated, then it might be that the surface is close to the edge of the cell, and there will be little information around the surface if the neighbouring voxels does not contain any information. By also allocating these, we can guarantee that the surface representation will contain enough information for the pose estimation.

2) Estimation the SDF and the Colors: When the voxel grid has been allocated, we need to estimate the distance between the voxels and the surface. Ideally, one would want that each voxel is assigned the smallest distance to the surface, however, this is very time consuming. Instead, we follow the idea by Curless and Levoy [3] of using the projected distance. That is, the distance between the 3D point and the surface along the viewing ray.

To estimate the projected distance, assuming the global rotation and translation is known, we transform the global coordinates $\mathbf{x}^G$ for each voxel to the camera frame of view by $\mathbf{x}^L = R^T(\mathbf{x}^G - \mathbf{t})$. When $\mathbf{x}^L$ is known, it is projected onto the image plane where we get the pixel $(u,v) = \pi(\mathbf{x}^L)$. Taking the z-coordinate z^L for the voxel in the camera frame and reading the depth $z = I_d^t(u,v)$ to the surface along the optical axis, the projected distance d is estimated as

$$d = z^L - z, \tag{12}$$

which is illustrated in Figure 4.

Since the projected distance is a rough approximation which can get arbitrary wrong, we follow the standard approach to reduce the impact of bad measurements by truncating the measured distance if $|d| > \delta$ for some threshold δ, i.e.

$$d_{trunc} = \begin{cases} -\delta & \text{if } d < \delta \\ d & \text{if } |d| \leq \delta. \\ \delta & \text{if } d > \delta \end{cases}$$

However, this is not enough to decrease the impact of bad measurements. We do also have a higher uncertainty when the voxel lies behind the surface. To handle this, we weight the measurements using the following weight function

$$w(d) = \begin{cases} 1 & \text{if } d \leq \epsilon \\ e^{-\sigma(d-\epsilon)^2} & \text{if } \epsilon < d < \delta. \\ 0 & \text{if } d \geq \delta \end{cases} \tag{13}$$

By using the weight and the truncated distance we can update the distance information in the voxel grid by computing the weighted average

$$D^{t+1} = \frac{W^t D^t + w^{t+1} d_{trunc}^{t+1}}{W^t + w^{t+1}} \tag{14}$$

$$W^{t+1} = W^t + w^{t+1}, \tag{15}$$

where D^t and W^t are the estimated distance and weight after the t first images and d_{trunc}^{t+1} and w^{t+1} are the estimated distance and weight for image $t+1$. By doing this for all voxels in the grid, we can fuse all the depth images into one representation only. The different thresholds δ and ϵ are evaluated in [1] and set to $\delta = 0.3m$ and $\epsilon = 0.025$.

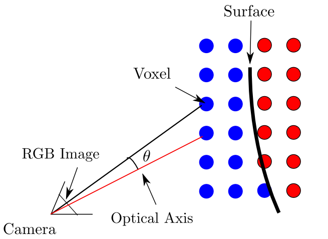


Fig. 5. By projecting each voxel onto the image plane we can extract the rgb vector in the color image and estimate the color for each voxel. The angle θ is used to weight the measurement.

We also need to estimate the color for each voxel in order to get a colourized 3D-model. The idea is to store an rgb vector which contains the color intensities in each voxel. To estimate the rgb vector for a voxel, we use that the pixels in the depth image and color image are in one-to-one correspondence. So when projecting the voxel onto the image plane, we get also the corresponding pixel coordinates for the color image I_c and we can extract the rgb vector

$$(r,g,b) = I_c(u,v), \tag{16}$$

as illustrated in Figure 5.

To fuse all the measurements the weighted average is computed, just like the distance measurements are fused.

As weight we choose the product

$$w_c = \cos(\theta) w(d) \tag{17}$$

where θ is the angle between the optical axis and the projected line between the surface point and the camera center, see Figure 5. This is so the parts which are looked more straight onto are getting a higher weight. Since we are measuring the colors for voxels which are not close to the surface, we also use the distance to the surface to weight the measurement by using the weight obtained from (13).

Thus for each new image and each voxel we estimate the rgb vector as

$$R = \frac{R^t W_c^t + r^{t+1} w_c^{t+1}}{W_c^t + w_c^{t+1}} \tag{18}$$

$$G = \frac{G^t W_c^t + g^{t+1} w_c^{t+1}}{W_c^t + w_c^{t+1}} \tag{19}$$

$$B = \frac{B^t W_c^t + b^{t+1} w_c^{t+1}}{W_c^t + w_c^{t+1}} \tag{20}$$

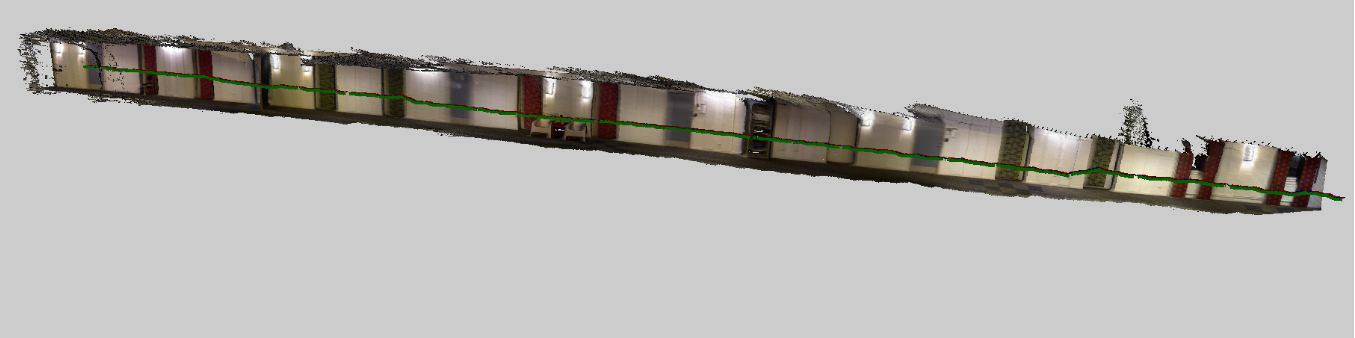


Fig. 6. A long corridor reconstructed, the scene is approximately 40 m long and the tracking need to be good over time to get a descent result. Note that the chairs looks good even though they are halfway in the scene, which indicates a consistent tracking. The camera path is plotted in the 3D model as well (red, green and blue lines indicates the x- y- and z-axis of the local camera frame), which shows no indication of loosing track.

where w_c is the weight of the certainty of the color measurement, R^t, G^t and B^t are the estimated colors for the first t images and W^t is the total weight after t images.

IV. Experiments

In this section we evaluate how well the proposed method works for different scenes. In particular we will show how it can handle scenes with no structure and also larger scenes where there is little structure.

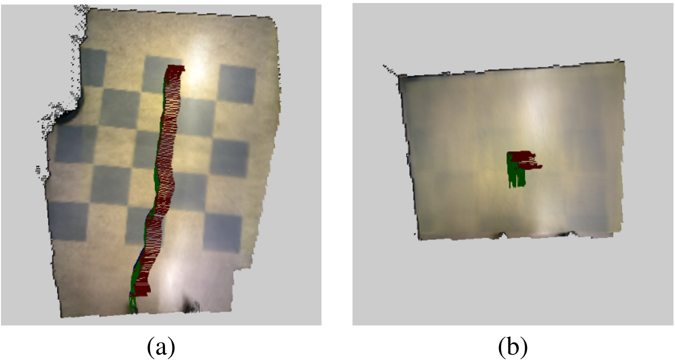
A. *Qualitative Results*

In this part we show 3D models acquired from our proposed algorithm, in particular we show the difference between the pure structure based algorithm from [1] and our new extension for a dataset with pure planar surfaces. All images are captured by using the Asus Xtion Pro Live sensor.

To test our method and see how well it works we do experiments by recording data from different environments. Our main hypothesis is that the tracking shall remain stable with no structure if there is enough texture in the scene. To test this we started with recording a small dataset with the camera only facing a floor with a regular pattern of blue squares. With pure planar structure, there is no unique minima for the error function which only takes structure into account. However, with our proposed error function (8), there should be a unique minimum. Comparing Figure 7 b) and Figure 7 a), we see that the estimated trajectory for the structure based method is a roughly stationary camera, but the trajectory for the combined structure- and color based method gives a clearly better estimation. In particular the pattern on the floor is easy to distinguish. Note that the edges on the squares are sharp, which indicates a good estimation of the camera pose.

Clearly, this shows that our proposed method is able to handle situations where there is no structure but there is texture. In Table I, it is also shown that our new method also can handle scenes with no texture but with structure as well. The extreme case which it cannot handle is scenes with no structure or texture.

By representing the signed distance function more effectively, we can also do larger scale reconstruction. In Figure 6 a reconstruction of a corridor is seen. This is quite a challenging scene because typically a corridor contains scenes with little structure as well as scenes with little texture, such as white walls. Moreover, the recorded scene is about 40 - 50 m long and together with the challenging scenes with plenty of planar surfaces, the tracking needs to be accurate and robust to not drift away.



(a)

(b)

Fig. 7. Difference between tracking which uses both structure and color (a) and tracking which uses only structure (b). Even though completely flat structure, the color based tracking works very good but the pure structure based does not. The red and green lines indicates the x- and y-axis of the camera frame. The z-axis is in blue and is facing the floor.

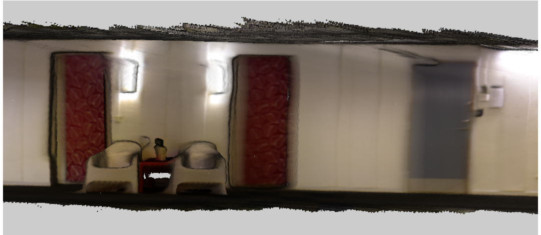


Fig. 8. Zoomming in on the scene we can clearly see that the pattern on the curtains are clearly distinguishable, which indicates a consistent tracking.

The resulting reconstruction is showed in Figure 6. The estimated pose is quite a straight path and there are no big jumps in the pose estimation. All this indicates that the tracking works well for larger scenes as well. In particular, if we look at Figure 8, we see that quality of the reconstructed chairs are good and that the pattern on the curtains are clearly distinguishable. If we look at Figure 6 again, we see that the chairs comes halfway in the scene, which indicates that the drift is small even after about 20 m.

TABLE I. THE ROOT-MEAN SQUARE ABSOLUTE TRAJECTORY ERROR (M) FOR DIFFERENT VALUES OF THE WEIGHT α IN (8).

Dataset	Weight α										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	Steinbruecker et al. [10]
fr1 teddy	0.083	0.0803	**0.079**	0.080	0.082	0.088	0.089	0.098	0.105	0.102	0.099	0.036
fr3 structure_no_texture	0.040	0.040	0.040	0.039	0.039	0.039	0.039	0.039	0.039	0.039	**0.038**	[2]-
fr3 no_structure_texture	1.36	0.038	**0.035**	0.037	0.039	0.040	0.042	0.043	0.044	0.045	0.047	-
fr3 no_structure_no_texture	0.756	0.796	0.754	0.700	0.657	0.609	0.584	0.568	**0.532**	0.538	**0.532**	-
fr1 floor	0.677	**0.374**	0.417	0.528	0.584	0.657	0.658	0.715	0.719	0.715	0.716	-
fr1 room	0.177	**0.171**	0.326	0.260	0.300	0.397	0.493	0.652	0.679	0.691	0.684	0.054
fr1 desk	0.037	0.033	0.032	0.032	0.031	0.031	**0.030**	**0.030**	**0.030**	0.031	0.031	0.021
fr1 desk2	0.063	**0.062**	0.066	0.067	0.073	0.084	0.083	0.084	0.096	0.115	0.127	0.027
fr1 360	0.117	**0.114**	0.115	0.120	0.136	0.138	0.150	0.163	0.168	0.177	0.314	0.073
fr1 long_office_househould	0.087	0.062	0.057	0.054	0.054	0.052	**0.051**	**0.051**	0.052	**0.051**	0.053	0.030

B. *Quantitative Results*

In Table I our proposed algorithm is evaluated using the benchmarks from [12]. To measure the impact of the color information, each dataset is evaluated ten times with different values for the weighting parameter α in (8). $\alpha = 0$ corresponds to the pure structure based method in [1] and $\alpha = 1$ gives the color error the same weight as the distance error. In particular, we evaluated our method on the benchmarks fr3 structure_no_texture and fr3 no_structure_texture, which corresponds to the two extreme cases for which we claim our algorithm is robust. The results shows clearly that for $\alpha = 0$, the tracking fails for fr3 no_structure_texture, but when color information is included the tracking works very well. Also the results for fr3 _structure_no_texture are convincing, which shows our proposed method is capable of handling both extreme scenarios. We also include the results we can find from the just published work by Steinbruecker et al. [10] to show what the state-of-art can achieve. They do clearly better on the benchmarks for which we have results, however, they use both graph optimization and loop-closure in their machinery.

V. CONCLUSION AND FUTURE WORK

The main contribution with this paper is that we can include both color and structure information from the global 3D-model represented as SDF. Experiments shows that this gives robustness for scenes with no structure but texture, as well as scenes with structure but lack of texture.

Evaluation on benchmarks confirms that we can handle both extreme scenarios. However, Steinbruecker et al. [10] clearly performs better on the benchmarks, it would be interesting to see how our method would perform if graph optimization was included in our method as well.

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] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. 2013.

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

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

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[6] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, 2011.

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

[8] C. Olsson, J. Ulén, and Y. Boykov. In defense of 3d-label stereo. In *CVPR*, 2013.

[9] F. Steinbruecker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)*, 2011.

[10] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, 2013.

[11] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Pattern Recognition (Proc. DAGM)*, Darmstadt, Germany, 2010.

[12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS*, 2012.

[13] T.M. Tykkälä, C. Audras, and A.I Comport. Direct iterative closest point for real-time visual odometry. In *Workshop on Computer Vision in Vehicle Technology at ICCV*, 2011.

[14] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J.B. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Karlsruhe, Germany, May 2013.

[2] No results available