Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions

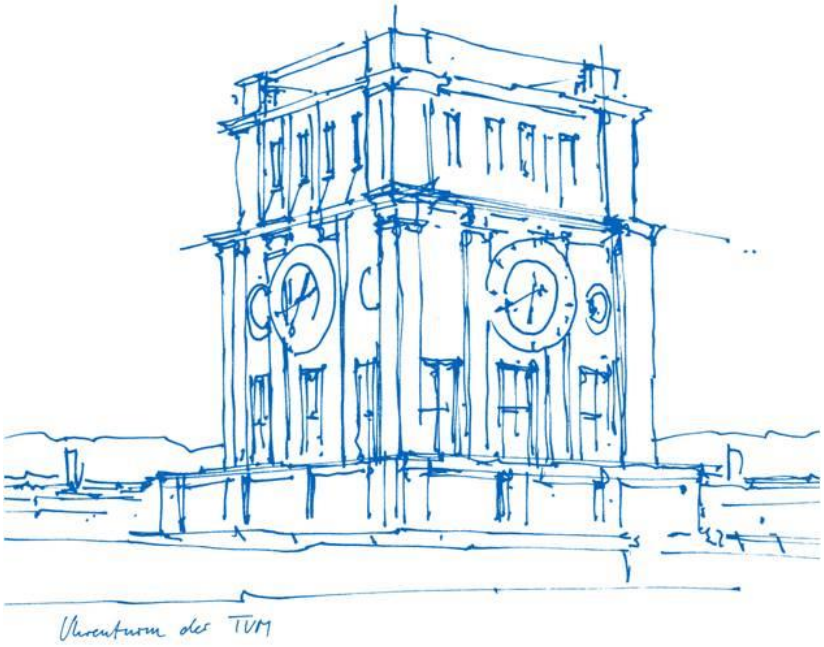
By Erik Bylow, Jurgen Sturmy, Christian Kerl, Fredrik Kahl, Daniel Cremers

Naveen Kumar Subramanian

M.Sc Computational Science and Engineering

Technische Universität München

April 2019



Abstract

- Novel method for real-time camera tracking and 3D reconstruction of static indoor environments using an RGB-D sensor.
- representing the geometry with a signed distance function (SDF), .
- Proved that benchmark data that our approach is more accurate and robust than the iterated closest point algorithm (ICP) used by KinectFusion

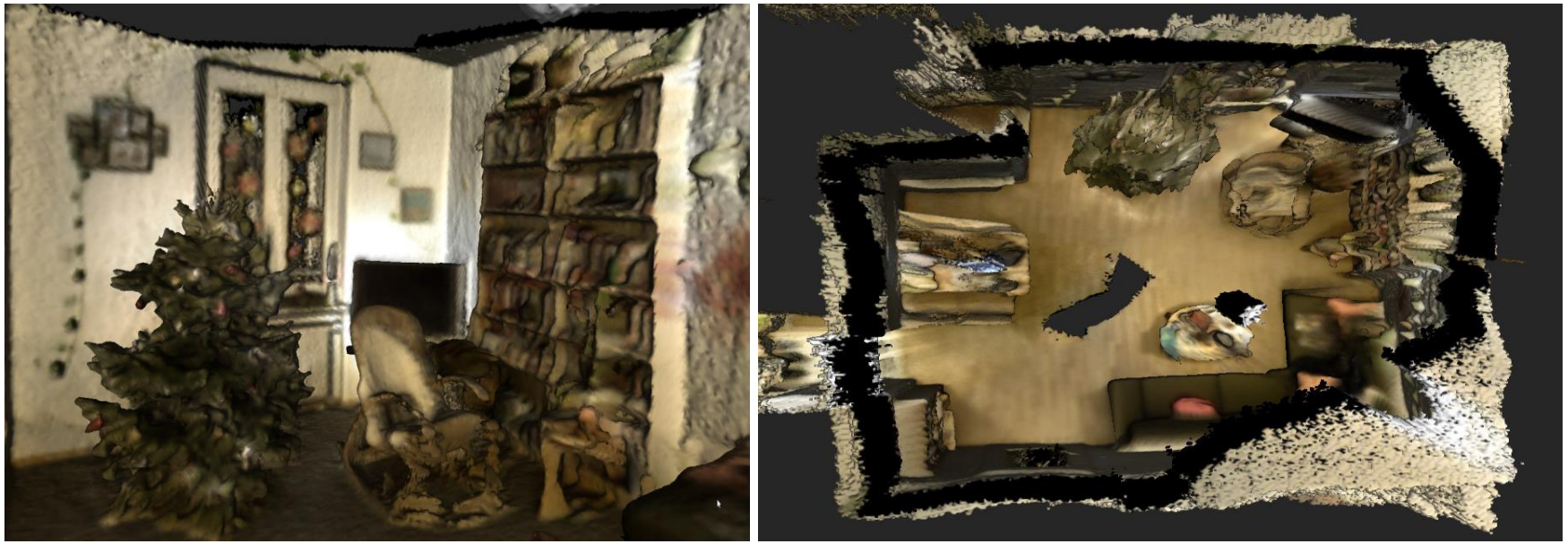
Main Contents Of The Paper

- A direct approach to camera tracking on SDFs
- Present a thorough evaluation of SDF based tracking and mapping on public benchmarks
- Compare the tracking performance to existing real-time solutions
- Additionally , We study the influence of alternative distance metrics , weighting functions

Introduction

- Simultaneous localization and mapping refers to both the estimation of the camera pose and mapping of the environment.

- Structure from motion (SfM) techniques from computer vision typically use images from a moving camera.



Reconstruction of a living room with a handheld sensor using our approach

Notations

- Rotation of Camera R ϵ SO(3)
- Translation t ϵ R^3
- At each time step, an RGB-D camera outputs a color and a depth image, to which we refer to by the functions

$$I_{RGB} : R^2 \rightarrow R^3 \text{ and } I_d : R^2 \rightarrow R^1$$

- The pinhole camera model with intrinsic parameters f_x, f_y, c_x and c_y corresponding to the focal length and the optical center. According to this model, a 3D point x = $(x, y, z)^T$ is projected onto the image plane by

$$\pi(x, y, z) = \left(\frac{f_x X}{z} + C_z, \frac{f_y y}{z} + c_z\right)^T$$

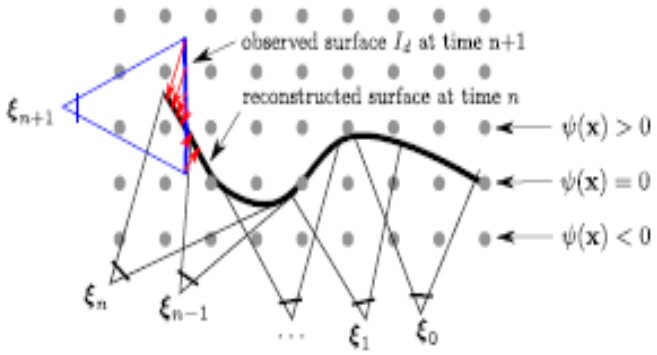
- we can reconstruct the 3D point corresponding to a pixel $(i, j)^T$ ϵ R^2 with depth z = I_d(i; j) by

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

Approach

1. Camera Tracking

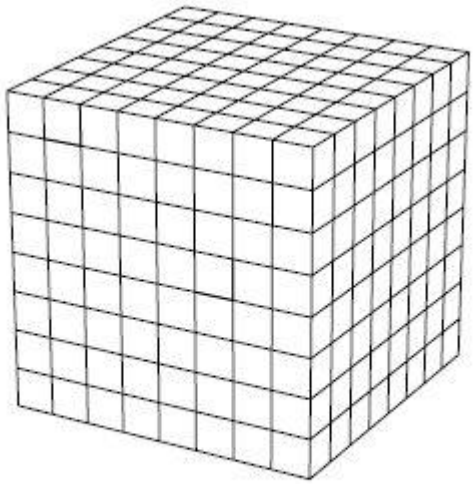
- Estimate the camera motion given an SDF and a depth image
- To find the camera rotation R and translation t such that all reprojected points from the depth image lie as close as possible to the zero-crossing in the SDF



Approach

2. **Representation of SDF**

- we represent the SDF using a discrete voxel grid of resolution m. We allocate two grids in memory, where one stores the averaged distances, and the second one stores the sum of all weights



8x8x8 *resolution orthogonal (uniform) voxel grid*

2.Representation of SDF

- Given a reconstruction volume of dimension width height depth, a world point x = $(x, y, z)^T \epsilon\ R^3$is mapped to voxel coordinates

$$\begin{pmatrix} i \\ j \\ k \end{pmatrix} = m \begin{pmatrix} x/\text{width} + 0.5 \\ y/\text{height} + 0.5 \\ z/\text{depth} + 0.5 \end{pmatrix}.$$

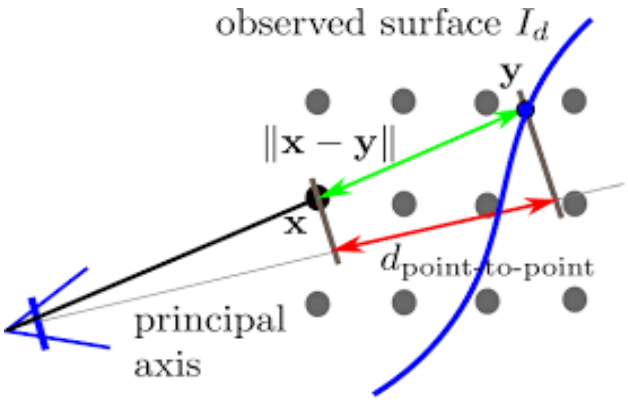
Since $(i, j, k)^T$ is generally non-integer, we determine the signed distance value (x) by tri-linear interpolation between ψ(x) the values of its eight integer neighbors.

Approach

3. Distance and Weighting Functions

- To determine which voxels have been observed by the depth camera and update their distances accordingly .

Projective Point-to-Point: the projective point-to-point distance as the difference of the depth of the voxel and the observed depth at $(i, j)^T$ negative values are assigned to voxels in front of the observed surface, and positive values to voxels behind.

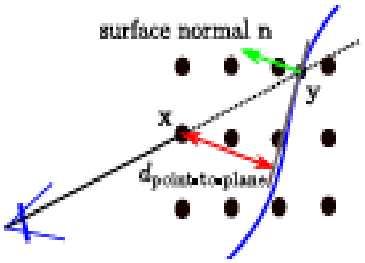


Approach

3. Distance and Weighting Functions

Projective Point-To-Plane: the point-to-point metric gets increasingly inaccurate the less the viewing angle is orthogonal to the surface **.** As a first step, we apply a bilateral filter to the depth image and compute the normals for all pixels. Given a voxel x, we compute its corresponding pixel coordinates (i; j) and read out the observed surface normal n(i; j). The point-to-plane distance can then be computed as

$$d_{point-to-plane}(x) := (y - x)^T n(i, j).$$



Approach

3. Distance and Weighting Functions

Truncation we truncate the projected distances d and apply a weighting term that blends out large distances, which makes the gradient of our SDF zero in regions that are far away from the estimated surface.

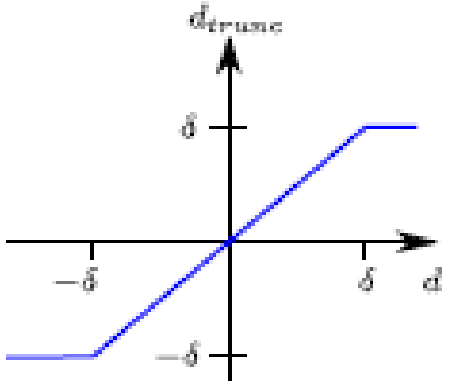
Weighting:. we employ a weighting function to give higher weights to voxels in front of the observed surface and lower weights to voxels behind.
Depending on the observation model of the depth sensor, different weighting functions can be used.

Approach

3. Distance and Weighting Functions

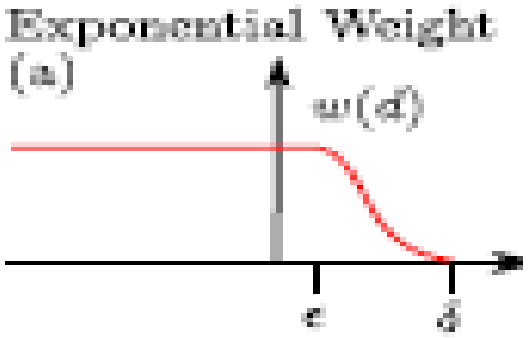
Truncation Function

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



Exponential Weight Function

$$W_{exp}(x) = \begin{Bmatrix} 1 & if\ \ d < \epsilon \\ e^{-\sigma(d-\epsilon)^2} & if\ \ d \geq \epsilon\ and\ d \leq \delta \\ 0 & if\ \ d < \delta \end{Bmatrix}$$



Approach

4. Data Fusion and 3D Reconstruction

- Goal to fuse all measurements to get a possible estimate of SDF ψ(x)
- All Distance measurements normally distributed
- By taking distance , the maximize the livelihood function ψ and taking negative logarithm we get

$$L(\psi) = \sum_{i=1}^{n} \frac{1}{2} w_i \quad (\psi - d_i)^2$$

- By taking the derivative and equating to zero , we get

$$\psi = \frac{\sum_{i=1}^{n} w_i d_i}{\sum_{i=1}^{n} w_i}$$

Approach

5. Meshing and Colorization

- Color texture using an additional voxel grid consisting of three channels R G B for the color and one additional channel for the color weights W_c
- we retrieve the observed color $(\boldsymbol{r}, \boldsymbol{g}, \boldsymbol{b})^{\boldsymbol{T}} = \boldsymbol{I_{RGB}}(\boldsymbol{i}, \boldsymbol{j})$
  from the RGB image and update the color estimate as the running average as

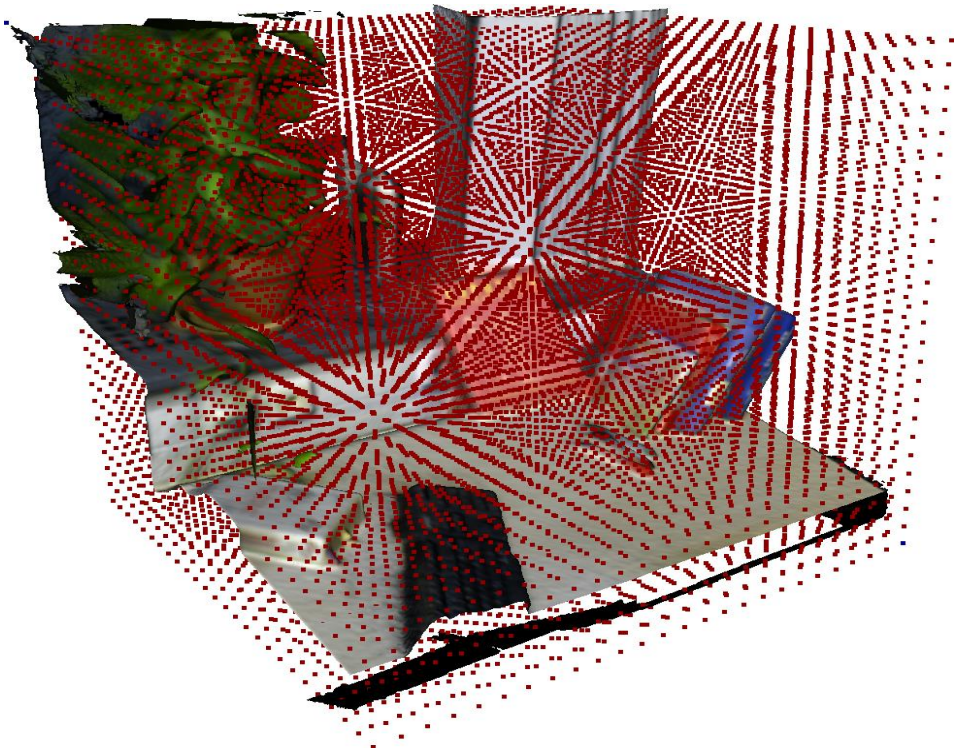
$$R \leftarrow \frac{W_c R + {W_c}^{n+1} r}{w_c + {W_c}^{n+1}}$$

and similarly for G and R , where ${W_c}^{n+1}$ is the weight of new measurement and it is given by

$${W_c}^{n+1} = W_{n+1} \cos \emptyset$$

$\emptyset$ is the angle between ray and principal axis

5. Meshing and Colorization

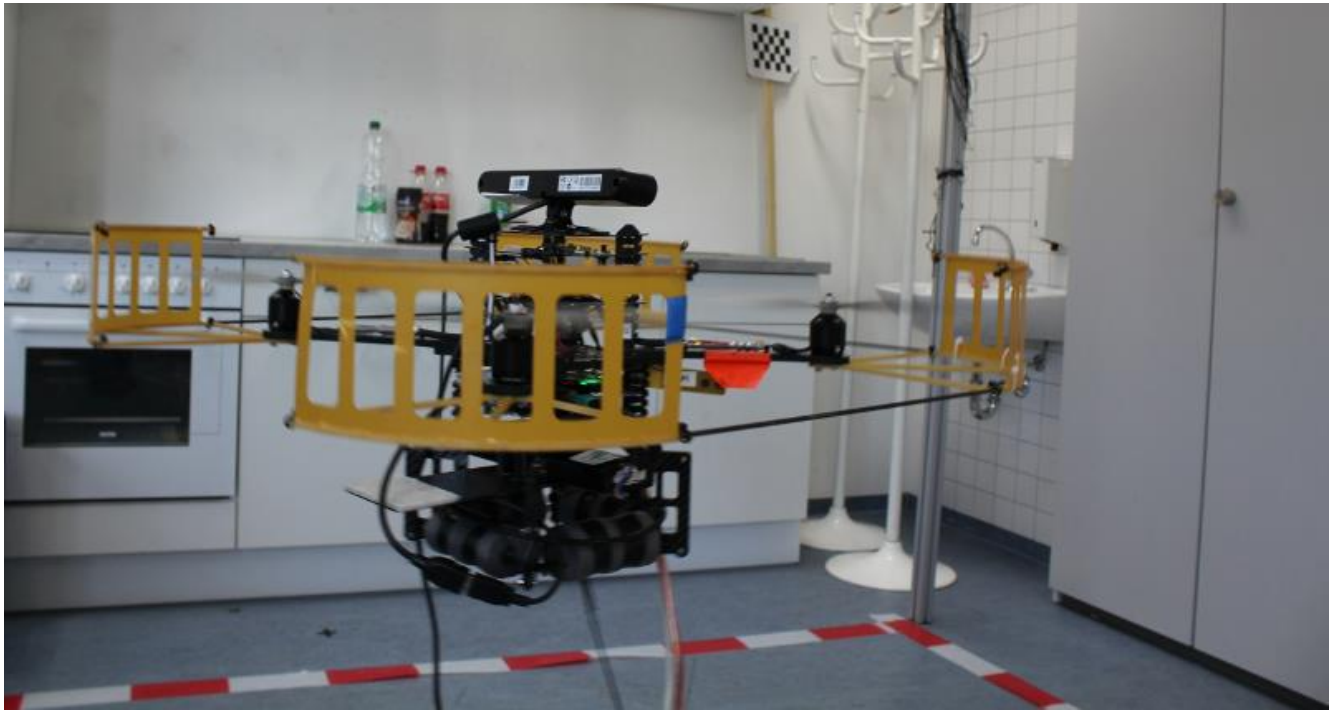


Visualization of the (downsampled) voxel grid underlying the reconstruction volume (m = 256).

Results and BenchMark

Experiment Setup:

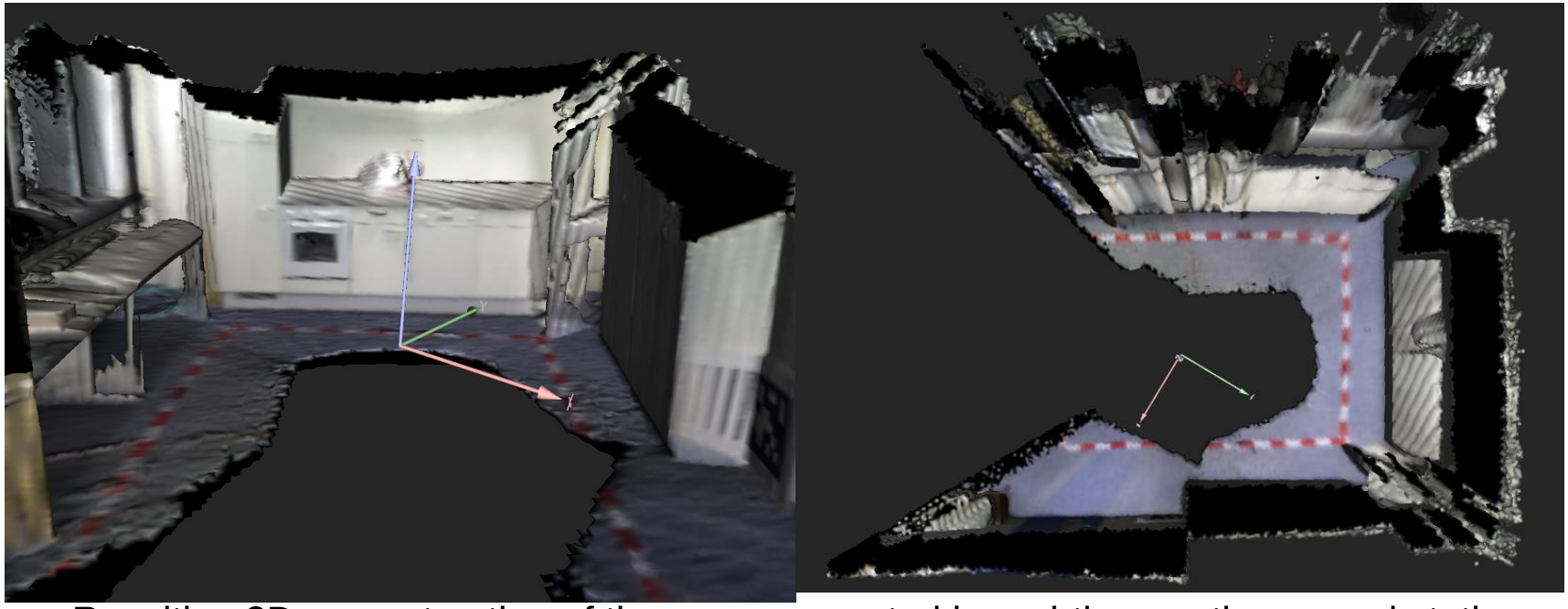
1. Handheld Asus Xtion Pro Live sensor Is used for Scanning
2. Laptop with a Quadro GPU from Nvidia gives live view of reconstructed Image



AscTec Pelican platform used.

Experiments

- This approach for 3D reconstruction from an autonomous quadrocopter (see Figure) equipped with an RGB-D camera.
- tracking and reconstruction were carried out in real-time on an external ground station
- This demonstrates that our technique is applicable for the navigation of quadrocopters and other robots.



Resulting 3D reconstruction of the room computed in real-time on the ground station

Benchmark

- Evaluated on TUM RGB-D BenchMark
- Compared with KinFu implementation and RGB-D SLAM

Parameters that are chosen are $\delta = 0.3\,m$ and $\epsilon = 0.025\,m$ and results are

TABLE I: The root-mean square absolute trajectory error for KinFu and our method for different resolutions, metrics and datasets. Also the result for RGB-D SLAM are presented.

Method	Res.	Teddy	F1 Desk	F1 Desk2	F3 Household	F1 Floor	F1 360	F1 Room	F1 Plant	F1 RPY	F1 XYZ
KinFu	256	0.156 m	0.057m	0.420 m	0.064 m	Failed	0.913 m	Failed	0.598 m	0.133 m	0.026 m
KinFu	512	0.337 m	0.068 m	0.635 m	0.061 m	Failed	0.591 m	0.304 m	0.281 m	0.081 m	0.025 m
Point-To-Plane	256	**0.072 m**	0.087 m	0.078 m	0.053 m	0.811 m	0.533 m	0.163 m	0.047 m	0.047 m	0.029 m
Point-To-Plane	512	0.101 m	0.059 m	0.623 m	0.053 m	0.640 m	0.206 m	0.105 m	**0.041 m**	**0.042 m**	0.026 m
Point-To-Point	256	0.086 m	0.038 m	**0.061 m**	**0.039 m**	0.641 m	0.420 m	0.121 m	0.047 m	0.047 m	**0.021 m**
Point-To-Point	512	0.080 m	**0.035 m**	0.062 m	0.040 m	**0.567 m**	**0.119 m**	**0.078 m**	0.043 m	**0.042 m**	0.023 m
RGB-D SLAM		0.111 m	0.026 m	0.043 m	0.059 m	0.035 m	0.071 m	0.101 m	0.061 m	0.029 m	0.013 m

Benchmark

Observations

- Our approach performs well due to the fact that KinFu looses much valuable information because of the down projection of the SDF
- In comparison to RGB-D SLAM we achieve often a similar performance in terms of Accuracy but its takes less computation time due to the fact

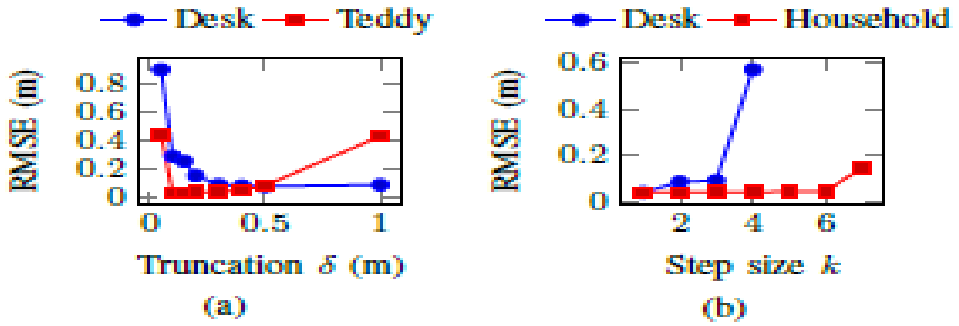


Fig. 8: (a) The choice of the truncation parameter δ depends on the average depth of the objects in the scene. For typical indoor scenes, $\delta = 0.3$ m is a good choice. (b) RMSE when using only every k-th frame (to emulate faster camera motions).

Summary

Limitations

- Since this approach only uses structure for tracking, it fails in cases where only co-planar surfaces are visible, such as a wall , but RGB-D SLAM uses texture
- additionally they did not exploit the color information during tracking

Conclusion:

- This method allows the quick acquisition of textured 3D models that can be used for real-time robot navigation
- For larger geometries, the combination of our method with a different SLAM solver would be interesting.
- Including and investigating methods that allow a more efficient representation of the 3D geometry will make it interesting.