Seminar Report of KinectFusion: Real-Time Dense Surface Mapping and Tracking

Wenliang Peng

Faculty of Informatics - Technische Universität München

Abstract

The paper ”KinectFusion: Real-Time Dense Surface Mapping and Tracking” was published by Microsoft Research and was published in 2011. The method proposed in this paper is a pioneering work based on a low-cost RGB-D camera for real-time reconstruction tracking and reconstruction. And it is an algorithm suitable for different lighting conditions, creating an important framework for 3D dense reconstruction. This paper shows that this method can accurately obtain tracking and mapping results in real-time in room-sized scenes using only commodity sensors and GPU hardware, which surpassed the solutions proposed using passive computer vision at that time.

1 Introduction

In computer vision, research on structure from motion (SFM) and multi-view stereo (MVS) has produced many compelling results. However, much of this work was not motivated by real-time applications. Simultaneous localization and mapping (SLAM) research focuses more on real-time unmarked tracking and real-time scene reconstruction based on monocular RGB camera input. However, although these systems perform real-time mapping, they are optimized for effective camera tracking, and the sparse point cloud models they produce can only perform basic scene reconstruction. With the advent of Microsoft Kinect, this highly cost-effective camera with a depth sensor can provide opportunities for SLAM and AR. This article propose a detailed method that allows the use of handheld Kinect depth sensors to perform real-time, dense volumetric reconstruction of complex room-sized scenes. The use of highly parallel general-purpose GPU (GPGPU) technology is at the core of all our design decisions. It allows tracking and mapping to be performed at the frame rate of the Kinect sensor and in constant time[3].

2 Method description

In this method, a dense reconstruction algorithm using depth maps is created to accurately reconstruct the 3D model in real-time. the algorithm pipeline is shown in Figure 1, including four parts, namely measurement, pose estimation, reconstruction update, and surface prediction.

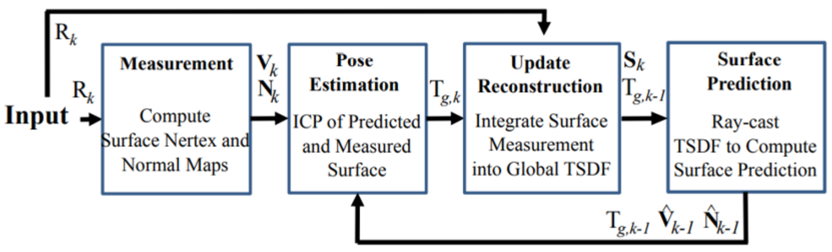
Input
R_k
R_k
Measurement
Compute Surface Nertex and Normal Maps
V_k
N_k
Pose Estimation
ICP of Predicted and Measured Surface
$T_{g,k}$
Update Reconstruction
Integrate Surface Measurement into Global TSDF
S_k
$T_{g,k-1}$
Surface Prediction
Ray-cast TSDF to Compute Surface Prediction
$T_{g,k-1}$ $\hat{V}_{k-1}$ $\hat{N}_{k-1}$

Figure 1: Overall system workflow[3].

2.1 Preliminaries

The pose of the real-time camera from the current frame k to the global frame g can be represented by a rigid body transformation matrix:

$$T_{g,k} = \left\{ \begin{pmatrix} R_{g,k} & \mathbf{t_{g,k}} \\ \mathbf{0} & 1 \end{pmatrix} : R_{g,k}^\top R_{g,k} = R_{g,k} R_{g,k}^\top = I_3, \mathbf{t_{g,k}} \in \mathbb{R}^3 \right\}, \tag{1}$$

A point $\mathbf{p_k} \in \mathbb{R}^3$ in the camera frame is transferred into the global co-ordinate frame via $\mathbf{p_g} = T_{g,k}\mathbf{p_k}$. And the constant camera calibration matrix K converts the points of the camera coordinate system into the points of the pixel coordinate system. The function $\mathbf{q} = \pi(\mathbf{p})$ performs the perspective projection of $\mathbf{p} \in \mathbb{R}^3 = (x, y, z)^\top$, including de-homogenization to obtain $\mathbf{q} \in \mathbb{R}^2 = (x/z, y/z)^\top$. This paper also uses dot notation to represent the homogeneous vector $\dot{\mathbf{u}} := (\mathbf{u}^\top|1)^\top$[3].

2.2 Surface measurement

In the k frame, the measurement includes the original depth map R_k, and the calibrated depth measurement value $R_k(\mathbf{u}) \in \mathbb{R}$ is provided on each image. The pixel is in the image domain $\mathbf{u} \in \mathbb{R}^2$, such that $\mathbf{p_k} = R_k(\mathbf{u})K^{-1}\dot{\mathbf{u}}$ is the measurement point in the reference k frame. The bilateral filter can be applied to the original depth map to obtain a discontinuity preserved depth map with reduced noise D_k,

$$D_k(\mathbf{u}) = \frac{1}{W_\mathbf{p}} \sum_\mathbf{q} \mathcal{N}_{\sigma_s}(||\mathbf{u} - \mathbf{q}||_2)\mathcal{N}_{\sigma_r}(||R_k(\mathbf{u}) - R_k(\mathbf{q})||_2)R_k(\mathbf{q}), \tag{2}$$

where $\mathcal{N}_\sigma(t) = \exp(-t^2\sigma^{-2})$ and $W_\mathbf{p}$ is a normalizing constant.

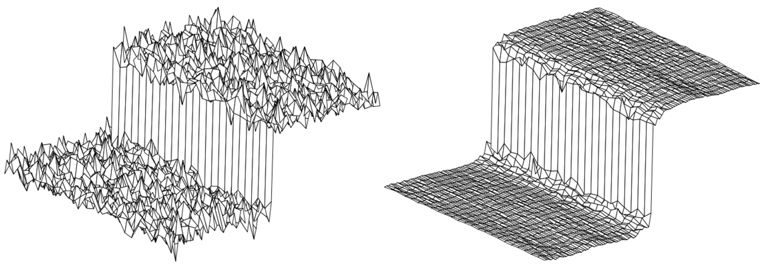

Figure 2: Result of denoising by bilateral filter[5].

The result of denoising is shown in the Figure 2. The bilateral filter can denoise the depth map while preserving the depth of the depth map.
The filtered depth values are back-projected to the frame of the reference to obtain the vertex map $\mathbf{V}_k$,

$$\mathbf{V}_k(\mathbf{u}) = D_k(\mathbf{u})K^{-1}\dot{\mathbf{u}} \tag{3}$$

The cross product between adjacent map vertices can be used to calculate the corresponding normal vector,

$$\mathbf{N}_k(\mathbf{u}) = \mathrm{v}[(\mathbf{V}_k(u+1,v) - \mathbf{V}_k(u,v)) \times (\mathbf{V}_k(u,v+1) - \mathbf{V}_k(u,v))], \tag{4}$$

where $\mathrm{v}[\mathbf{x}] = \mathbf{x}/||\mathbf{x}||_2$.
The vertex and normal map pyramid need to be built as a multi-scale representation of the surface measurement. Setting the bottom depth map pyramid level equal to the original bilateral filtered depth map, the sub-sampled version D^{l+1} is computed from D^l by block averaging followed by sub-sampling to half the resolution. Then each level in a vertex and normal map pyramid $\mathbf{V}^{l\in[1\ldots L]}$, $\mathbf{N}^{l\in[1\ldots L]}$ is computed with Equations 3 and 4 using the corresponding depth map level.

2.3 Pose estimation

The key concepts of the standard ICP algorithm can be summarized in two steps[4]:

1. Calculate the correspondence between two point sets.
2. Calculate the transformation between corresponding points to minimize the distance.

Because the corresponding relationship of the point sets may not be completely correct, we need to iteratively obtain the new corresponding relationship to calculate the desired transformation.
There are many ways to calculate the distance of the ICP algorithm. In KinectFusion, we use the global point-to-plane distance. Point-to-plane distance considers sum of the squared distance between each source point and the tangent plane at its corresponding destination point (see Figure 3). Compared with the point-to-point distance, point-to-plane distance considers the local structure of the point cloud, which has higher accuracy and is not easy to fall into the local optimum. if $\mathbf{s_i}$ is a source point, $\mathbf{d_i}$ is the corresponding destination point, and $\mathbf{n_i}$ is the unit normal vector at $\mathbf{d_i}$. The global point-plane ICP formula is as follows[2],

$$\mathbf{M}_{opt} = \arg\min_{\mathbf{M}} \sum_i ((\mathbf{M} \cdot \mathbf{s_i} - \mathbf{d_i}) \cdot \mathbf{n_i})^2, \tag{5}$$

where $\mathbf{M}$ and $\mathbf{M}_{opt}$ are 4×4 3D rigid-body transformation matrices.
Using the ICP algorithm, we can estimate the rotation and translation of the current frame. Applying the Equation 5 of global point-plane ICP, we can obtain the following expression,

$$\mathbf{E}(T_{g,k}) = \sum ||(T_{g,k}\dot{\mathbf{V}}_k(u) - \hat{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}}))^\top \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})||_2, \tag{6}$$

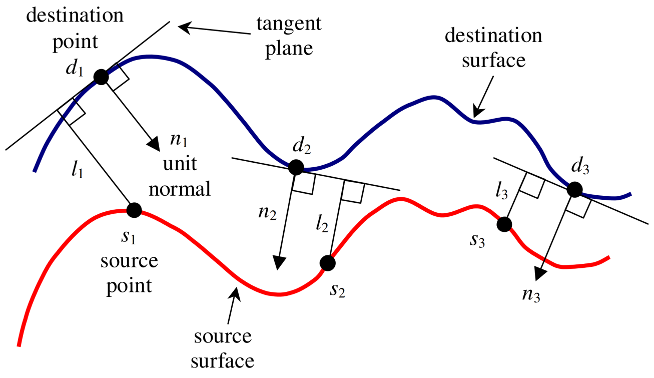

Figure 3: Point-to-plane error between two surfaces[2].

where $\mathbf{u}$ is the pixel of the current frame, $\hat{\mathbf{V}}^g_{k-1}$, and $\hat{\mathbf{N}}^g_{k-1}$ are the vertex map and normal map of the surface prediction in the global model built from all $k-1$ frames. $\hat{\mathbf{u}} = \pi(K\tilde{T}_{k-1,k}\dot{\mathbf{V}}_k(\mathbf{u}))$ is the result of projecting the pixel of the current frame to the $k-1$ frame through rotation and translation using an estimate for the frame-frame transformation $\tilde{T}^z_{k-1,k} = \tilde{T}^{-1}_{g,k-1}\tilde{T}^z_{g,k}$. The frame-frame transformation matrix $\tilde{T}^z_{k-1,k}$ here contains the current transformation matrix $\tilde{T}^z_{g,k}$ that we need to solve, so the correspondence between pixels $\mathbf{u}$ and $\hat{\mathbf{u}}$ may not be correct. We need to iteratively solve the transformation matrix, and z means the number of iterations.
Although we have expressed the function to be optimized, this function is not in the form of linear least squares and it is difficult to optimize. Now, the objective function of ICP needs to be transformed into the form of linear least squares. The camera pose is obtained continuously through incremental matrix $\tilde{T}^z_{inc}$ by iteration. In each iteration, the pose matrix is continuously multiplied by the incremental matrix($\tilde{T}^z_{g,k} = \tilde{T}^z_{inc}\tilde{T}^{z-1}_{g,k}$), and the new matrix continuously approaches the real camera pose. The increment matrix can be written in the following form with the small-angle assumption,

$$\tilde{T}^z_{inc} = \left[\tilde{R}^z|\tilde{\mathbf{t}}^z\right] = \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix}. \tag{7}$$

If the rigid body transformation matrix is written in the form of twist coordinate, the first term($T_{g,k}\dot{\mathbf{V}}_k(u)$) with respect to the iteration z of the Equation 6 can be written as a linear relationship to $\mathbf{x} = (\beta, \gamma, \alpha, t_x, t_y, t_z)^\top$,

$$\tilde{T}^z_{g,k}\dot{\mathbf{V}}_k(\mathbf{u}) = \tilde{R}^z\tilde{\mathbf{V}}^g_k(\mathbf{u}) + \tilde{\mathbf{t}}^z = \mathbf{G}(\mathbf{u})\mathbf{x} + \tilde{\mathbf{V}}^g_k(\mathbf{u}), \tag{8}$$

where the 3×6 matrix $\mathbf{G}$ is formed with the skew-symmetric matrix form of $\tilde{V}^g_k(\mathbf{u})$:

$$\mathbf{G}(\mathbf{u}) = \left[\tilde{V}^g_k(\mathbf{u})_\times|\mathbf{t}_{3\times3}.\right] \tag{9}$$

and $\tilde{V}^g_k(\mathbf{u}) = \tilde{T}^{z-1}_{g,k}\dot{\mathbf{V}}_k(\mathbf{u})$.
An iteration is obtained by solving:

$$\min_{\mathbf{x}} \sum ||E||^2_2 \tag{10}$$

$$E = \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})^\top(\mathbf{G}(\mathbf{u})\mathbf{x} + \tilde{\mathbf{V}}^g_k(\mathbf{u}) - \tilde{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}})) \tag{11}$$

Finally, the objective function E is expressed in a linear form, and it satisfies the First-order optimality condition, $\mathbf{x}$ can be obtained as follows:

$$\sum(\mathbf{A}^\top\mathbf{A})\mathbf{x} = \sum\mathbf{A}^\top\mathbf{b}, \tag{12}$$

$$\mathbf{A}^\top = \mathbf{G}^\top(\mathbf{u})\hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}}), \tag{13}$$

$$\mathbf{b} = \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})^\top(\tilde{\mathbf{V}}^g_k(\hat{\mathbf{u}}) - \tilde{\mathbf{V}}^g_{k-1}(\mathbf{u})). \tag{14}$$

The data association and pose minimisation is embedded into a coarse to fine framework using the bottom 3 levels of a vertex and normal map pyramid. We iterate for a maximum of z_{max} = [4,5,10] iterations in levels [3,2,1] respectively, starting with the coarsest level 3. In this paper, the problem is highly non-convex. Our algorithm for pose estimation can easily fall into a local optimum. Therefore, a good initialization is very useful to obtain the global optimum. The image pyramid is a scheme from coarse to fine, which can provide good initialization for optimization from coarse level to fine level, which can help us solve the initialization problem.

2.4 Surface reconstruction update

In the third part we need to update the reconstruction with the pose, here we use TSDF(truncated signed distance function). It is a representation method of the 3D map. It uses a large volume. The volume is composed of many small voxels. Each voxel corresponds to a point in space. Two components are stored at each location of the TSDF: the current truncated signed distance value $F_k(\mathbf{p})$ and a weight $W_k(\mathbf{p})$,

$$\mathbf{S}_k(\mathbf{p}) \mapsto [F_k(\mathbf{p}), W_k(\mathbf{p})]. \tag{15}$$

Through the following formula,

$$F_k(\mathbf{p}) = \Psi(\lambda^{-1}||\mathbf{t}_{g,k} - \mathbf{p}||_2 - R_k(\mathbf{x})), \tag{16}$$

$$\lambda = ||K^{-1}\dot{\mathbf{x}}||_2, \tag{17}$$

$$\mathbf{x} = \lfloor \pi(KT^{-1}_{g,k}\mathbf{p}) \rfloor, \tag{18}$$

$$\Psi(\eta) = \begin{cases} \min(1, \frac{\eta}{\mu})\mathrm{sgn}(\eta) & \text{iff } \eta \geq -\mu \\ null & otherwise \end{cases} \tag{19}$$

where $\lfloor\cdot\rfloor$ is a nearest neighbour lookup, instead of interpolating the depth value, $1/\lambda$ converts the ray distance to $\mathbf{p}$ to a depth, Ψ performs the SDF truncation, the weight $W_k(\mathbf{p})$ is related to the uncertainty of surface measurement, we can calculate the TSDF value of each voxel. If the value is greater than or less than the threshold, it will be truncated to 1 or -1. Only the TSDF value near the surface will be accurately represented as shown in Figure 4 .

The global fusion of all depth maps in the volume is formed by the weighted average of all individual TSDF, which can be seen as de-noising the global TSDF from multiple noisy TSDF measurements:

$$\min\sum||W_{R_k}F_{R_k} - F||_2. \tag{20}$$

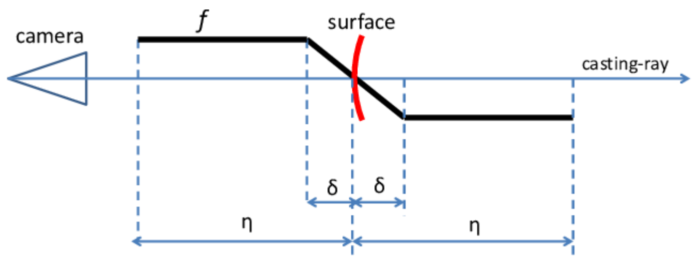

Figure 4: Illustration of the truncated signed distance function[1].

Since our focus is on real-time sensor tracking and surface reconstruction, we continue to integrate the new TSDF measurement into the global TSDF measurement:

$$F_k(\mathbf{p}) = \frac{W_{k-1}(\mathbf{P})F_{k-1}(\mathbf{P}) + W_{R_k}(\mathbf{P})F_{R_k}(\mathbf{P})}{W_{k-1}(\mathbf{P}) + W_{R_k}(\mathbf{P})}, \tag{21}$$

$$W_k(\mathbf{p}) = W_{k-1}(\mathbf{P}) + W_{R_k}(\mathbf{P}). \tag{22}$$

2.5 Surface prediction

In the last step, we can construct a new global vertex map $\hat{\mathbf{V}}_k^g(\mathbf{u})$ and normal map $\hat{\mathbf{N}}_k^g(\mathbf{u})$ for the second step based on the current global TSDF reconstruction. This paper uses the ray casting method. The ray, $T_{g,k}K^{-1}\dot{\mathbf{u}}$, starts from the minimum depth of each pixel $\mathbf{u}$ and follows direction until it encounters the zero-crossing($F_k(\mathbf{p}) = 0$), and the value of TSDF changes from positive to negative. We find the points of the surface based on the zero-crossing position and added them to the global vertex map $\hat{\mathbf{V}}_k$.
and the surface normal for the associated pixel $\mathbf{u}$ along which $\mathbf{p}$ was found can be computed directly from F_k using a numerical derivative of the SDF:

$$R_{g,k}\hat{\mathbf{N}}_k = \hat{\mathbf{N}}_k^g(\mathbf{u}) = \mathrm{v}[\nabla F(\mathbf{p})], \nabla F = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right]^\top \tag{23}$$

3 Experiments and results

In our experiments, the Kinect sensor is placed in a fixed position to observe the desktop scene installed on the turntable. Then the depth data is captured in approximately 19 seconds, so that the turntable rotates a full circle, resulting in $N = 560$ frames. All parameters of the system are kept constant using a reconstruction resolution of 256^3 voxels unless stated otherwise.
The N frames of depth data captured were then processed in each of the following ways:

1. Frames $1...N$ were fused together within the TSDF using sensor pose estimates obtained with frame-to-frame ICP implementation.

2. Frames $1...L, L < N$ were fed through our standard tracking and mapping pipeline. Here, sensor pose estimates are obtained by the full frame-model ICP method.

3. Frames $1 \ldots N$ were fed through our standard tracking and mapping pipeline. Sensor pose estimates are obtained by frame-model ICP.

4. Frames $1 \ldots N$ were fed repeatedly for $M = 4$ loops to the standard tracking and mapping pipeline. Sensor pose estimates are obtained by frame-model ICP.

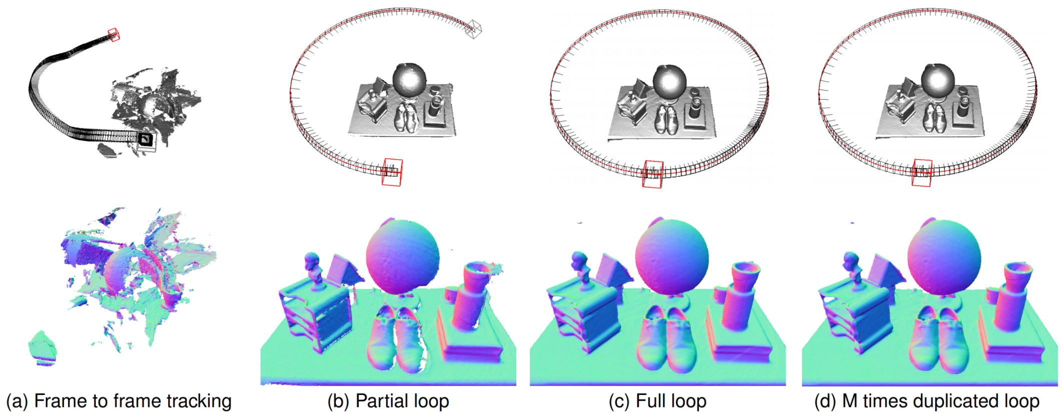

Figure 5: Circular motion experiment of our system as the sensor orbits a table[3].

(a), frame-to-frame tracking, the pose of each new frame is estimated by just the last frame. Rapid accumulation of errors results in the non-circular trajectory and poor reconstruction is apparent. (b)(c)(d) show our full frame-to-model tracking approach. From (b)(c)(d), we can obtain better 3D reconstruction results using our frame-to-model tracking approach.

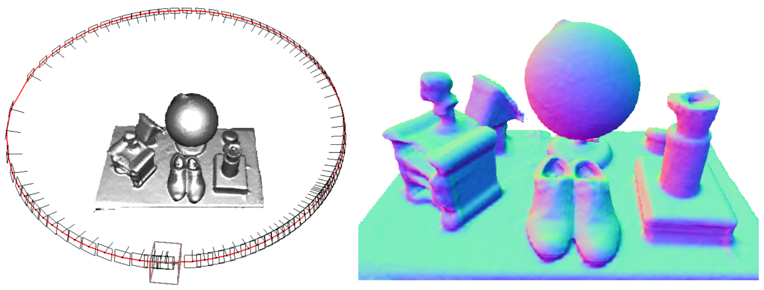

Figure 6: A reconstruction result using $\frac{1}{64}$ the memory of the Figure 5, and using only every 6_{th} sensor frame[3].

Figure 6 shows the reconstruction result where the the N frames are sub-sampled in time to use every 6_{th} frame, and 64 times less GPU memory is used by reducing the reconstruction resolution to 64^3. The computing and storage resources required to build the model have been greatly reduced, but the reconstruction quality has not dropped significantly.

4 Discussion / Conclusion

Compared with the frame-to-frame model, our frame-to-model system can obtain high-quality reconstruction results through the above experiment, even in the case of low resolution. But our model page has a main failure, our model is difficult to reconstruct a very large plane. The reason is that we use the global point-plane ICP algorithm to estimate the pose of the camera. one possible solution is to add photometric cost.
In this work, we implemented a real-time tracking and mapping method from frame to model. The key concepts in our real-time tracking and mapping system are:

- Up-to-date surface representation fusing all registered data from previous scans.
- Accurate and robust tracking of the camera pose by aligning all depth points with the complete scene model.
- Parallel algorithms for both tracking and mapping, taking full advantage of commodity GPU processing hardware.

A interesting direction is to reconstruct a large model. The current method is only suitable for drawing small rooms. However, rebuilding large models (such as entire buildings) adds many other challenges. The current dense volume representation will require too much memory and more space, and will cause inevitable drift.

References

[1] Zhuoliang Kang and Gérard G. Medioni. Fast dense 3d reconstruction using an adaptive multiscale discrete-continuous variational method. *IEEE Winter Conference on Applications of Computer Vision*, pages 53–60, 2014.

[2] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. 01 2004.

[3] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[4] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. Generalized-icp. 06 2009.

[5] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, 1998.