Brief Analysis of KinectFusion: Real-Time Dense Surface Mapping and Tracking

Tian Jin

Informatik - Technische Universität München

Abstract

Newcombe et al. introduced a system, KinectFusion, to realize real-time robust mapping and tracking regardless of lightning conditions in 2011. Kinect is a device with infrared emitter and sensor, through which depth map of the target scene could be obtained. By using the depth map as an input, the system could perform accurate 3D reconstruction. Iterative closest point (ICP) algorithm is used for camera pose estimation. Thus, all data obtained from the Kinect will be fused into the reconstruction model. In KinectFusion, space could be described by volumetric representation. The space is divided into small squares, and each square is called a voxel. This paper will focus on the methods and algorithms used in this system. After a brief introduction on 3D reconstruction and KinectFusion, the detailed concept in each step will be discussed. Finally, experiments and results will be presented with analysis of advantages and disadvantages.

1 Introduction

3D reconstruction is a technique to obtain the appearance and shape of real objects, then build a global model of it. As a popular topic in the computer vision domain, it provides more comprehensive geometric information then 2D images, and it could be applied in augmented reality and virtual reality application as a connection between 2D and 3D.

Conventional methods of 3D reconstruction include structure from motion (SFM) and multi-view stereo (MVS). Basically, according to a series of images which was captured at different locations, the 3D model could be obtained offline. Although several researches on them have shown remarkable results, those methods are hard to be used in real-time application.

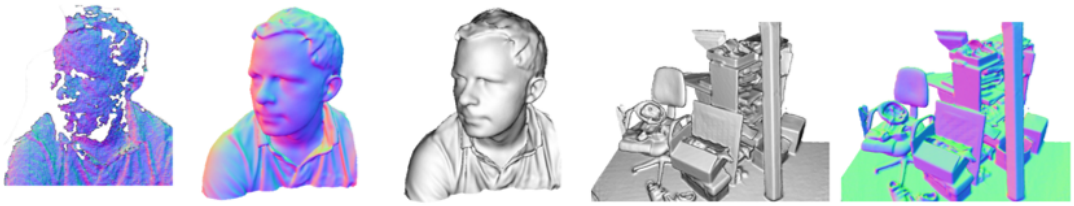

Figure 1: Example input and output[4]

In 2011, Newcombe et al. proposed a new system, KinectFusion, which made real-time 3D reconstruction, using only a low-cost and hand-held RGBD camera, come true. In this system, according to the raw depth map obtained from a Kinect device, the complex and arbitrary indoor scenes could be reconstructed with high accuracy in real time. As an example shown in Figure 1, the left image is an input to this system. With the Kinect depth camera only, a normal map (color) and Phong-shaded renderings (grey scale) could be generated.

In this paper, we will briefly go through the methods used in KinectFusion. Using only depth information, the system could track 6 degrees-of-freedom camera pose continuously (tracking), and all data obtained will be fused into one single global model rather than capturing features and using sub-data-set (mapping). As a result, the system successfully mitigates the influence of lightning conditions and have higher accuracy. Generally, the user only holds a Kinect device and moves around the target scene, a smooth and up-to-date 3D surface model will be reconstructed. With highly parallel general purpose GPU(GPGPU),both camera pose and the model could be updated

at the Kinect sensor's frame rate (30Hz). A series of experiments and results will also be presented and analyzed.

2 Method Description

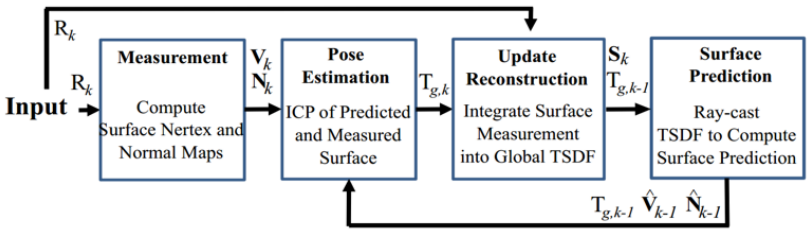

Figure 2: Overall system work flow [4]

Figure 2 shows an overall work flow of KinectFusion. The system is composed of four steps. Firstly, the system deals with the raw depth map streamed from the Kinect sensor in order to generate a dense vertex map and normal map pyramid. Then, according to the alignment between the current sensor measurement and the predicted surface, the camera pose could be estimated. After that, with the estimated camera pose, the surface measurement will be integrated into the scene model with truncated signed distence function (TSDF) representation. Finally, The TSDF value will be used to compute surface prediction. In the following sessions, each step will be discussed in details.

2.1 Preliminaries

- The 6DOF camera pose is represented by a rigid body transformation matrix at time k:

$$T_{g,k} = \begin{bmatrix} R_{g,k} & \mathbf{t}_{g,k} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{SE}_3 \quad where \quad \mathbb{SE}_3 := \{R, \mathbf{t} \mid R \in \mathbb{SO}_3, \mathbf{t} \in \mathbb{R}^3\}. \tag{1}$$

  R denotes rotation transformation and t denotes translate transformation. As the scene's size should not be changed, the scale parameter is set to 0 and 1. Then, a point $\mathbf{p}_k$ in the camera frame could be transferred in to the global frame:

$$\mathbf{p}_g = T_{g,k}\mathbf{p}_k \tag{2}$$

- $\mathbf{K}$ is used to denote the camera calibration matrix which transforms points on the sensors plane into image pixels.

- Dehomogenisation:

$$\mathbf{q} = \pi(\mathbf{p}) \quad where \quad \mathbf{p} \in \mathbb{R}^3 = (x, y, z)^T, \quad \mathbf{q} \in \mathbb{R}^2 = (x/z, y/z)^T \tag{3}$$

- Homogeneous vectors:

$$\dot{\mathbf{u}} = (\mathbf{u}^T \mid 1)^T \tag{4}$$

2.2 Surface measurement

Surface measurement is a pre-processing stage. Figure 3 shows the work flow of this step. $\mathbf{u}$ is used to denote an image pixel, such that $\mathbf{u}$=$(u,v)^T$.Firstly, bilateral filter is applied to raw depth map R_k to reduce noise. To recalculate the depth value of a specific pixel, weights are assigned to the surrounding pixels within a certain range. After weighted average, we will get the depth value of the current point with noise reduced. W_s refers to spatial proximity, and W_r refers to the

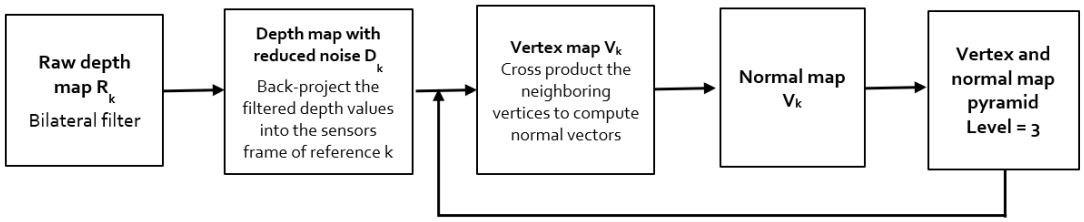

Figure 3: Surface measurement work flow

brightness similarity. In other words, the closer the surrounding pixel is, the more correlated the two pixels are, the bigger the value of W_s is; the less the difference between two pixels' brightness, the more correlated the two pixels are, the bigger the value of W_r is.

$$D_k(\mathbf{u}) = \frac{1}{W_\mathbf{p}} \sum W_s W_r R_k(\mathbf{q}) \tag{5}$$

By back-projecting the filtered depth values into the sensors frame, the vertex map is generated:

$$\mathbf{V}_k(\mathbf{u}) = D_k(\mathbf{u}) \mathbf{K}^{-1} \dot{\mathbf{u}} \tag{6}$$

As normal vectors could be calculated by the cross product of vectors, we can simply get the normal map by computation between neighbouring vertices, $v[x]$ is the normalizing step:

$$\mathbf{N}_k(\mathbf{u}) = v[(\mathbf{V}_k(u+1,v) - \mathbf{V}_k(u,v)) \times (\mathbf{V}_k(u,v+1) - \mathbf{V}_k(u,v))] \quad where \quad v[x] = x / \parallel x \parallel_2 \tag{7}$$

However, a three-level vertex and normal map pyramid is needed in this system. Firstly, we set the bottom level to the original filtered depth map (D_k). Then, from bottom to top, we half the resolution for each level by block averaging and sub-sampling. After that, we get a depth map pyramid. For each level, we could compute the vertex map and normal map using equation 6 and 7. Using a pyramid rather than a single depth map, the camera pose could be computed from coarse-to-fine which speed up the computation. Furthermore, the pyramid helps to get a more accurate result. If we conduct computation directly using the original filtered depth map, there may be more errors during the alignment process. The bigger the data set, the more details it contains, also, the more similar features it may have for different locations which results in wrong alignment.

2.3 Mapping as Surface Reconstruction

Surface reconstruction is the global scene fusion process, it fused all data of each frame into a global scene model consecutively, using truncated signed distance function (TSDF) as representation.

2.3.1 TSDF Representation

There are two values stored in a voxel in a TSDF model $\mathbf{S}_k(\mathbf{p})$, one is distance value$\mathbf{F}_k(\mathbf{p})$, another is weight value $\mathbf{W}_k(\mathbf{p})$. As Figure 4 shows, on the surface the distance value stored as 0, the visible side stored as positive value, and the invisible side as negative value. On visible side, the nearer the voxel to the sensor, the bigger the value would be; On the other side, the farther the voxel to the sensor, the smaller the value would be. Besides, as for surface reconstruction, only the voxels near the surface are meaningful. Thus, a truncated value is set. Every voxel whose distance to the surface is bigger than this value, will be stored as 1 or -1, as Figure 5 shows. The weight value is related to the uncertainty of measurement.

2.3.2 Calculation

Suppose that the uncertainty could be truncated, the true value lies within $\pm\mu$ of the measured value.

$$\mathbf{F}_{R_k}(\mathbf{p}) = \Psi(\lambda^{-1} \parallel \mathbf{t}_{g,k} - \mathbf{p} \parallel_2 - R_k(\mathbf{x})) \tag{8}$$

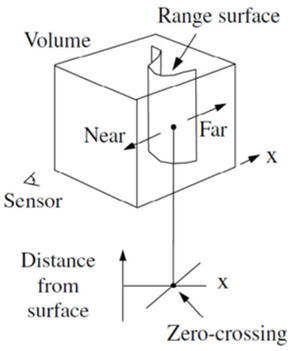

Figure 4: TSDF model [1]

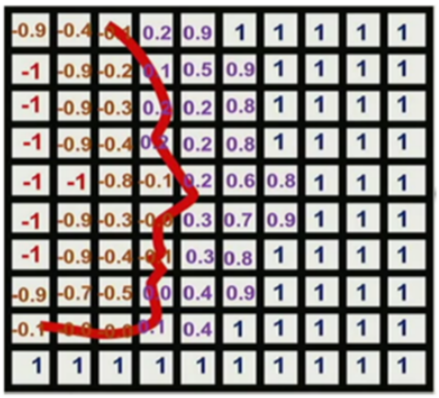

Figure 5: TSDF representation example [2]

$$\lambda = \| \mathbf{K}^{-1}\dot{\mathbf{x}} \|_2 \tag{9}$$

$$\mathbf{x} = \lfloor \pi(\mathbf{K}T_{g,k}^{-1}\mathbf{p}) \rfloor \tag{10}$$

$$\Psi(\eta) = \begin{cases} min(1, \frac{\eta}{\mu})sgn(\eta) & iff \eta \geq -\mu \\ null & otherwise \end{cases} \tag{11}$$

$\mathbf{p}$ is a 3D global point, while $\mathbf{x}$ is the corresponding 2D pixel point. λ is a normalization factor. According to Equation 8, we compute the ray from the camera center to the 3D point p (approximates the z coordinate),then minus the original depth data $\mathrm{R}_k(\mathbf{x})$. After normalization, we get the distance value of voxel $\mathbf{p}$. The weight value should be proportional to $cos\theta/\mathrm{R}_k(\mathbf{x})$.

Finally, the global TSDF could be updated by weighted averaging between previous TSDF model and current TSDF value. Although $\mathrm{W}_k(\mathbf{p})$ is proportional to the previous equation, we can simply let $\mathrm{W}_k(\mathbf{p}) = 1$ which provides good results practically. We may also set a truncated value of weight as if the value is very small, the distance value would be meaningless.

2.4 Surface prediction from ray casting the TSDF

As discussed previously, when $\mathrm{F}_k(\mathbf{p}) = 0$, the voxel $\mathbf{p}$ will be on the surface. Thus, a method, called ray casting, could be used to estimate the surface. Basically, the ray starts from the camera, goes against the light, until the distance value stored in each voxel goes from positive to negative or vise versa. According to the definition of TSDF representation, surface must be between those two voxels.

2.5 Sensor pose estimation

Iterative closest point (ICP) algorithm [5] is used to compute the rigid body transformation. In this system, frame-to-model ICP is used rather than general frame-to-frame ICP. In other words, we won't find the corresponding point cloud through two consecutive frames. Instead, we project the point a of current measured surface to the previous camera pose, the point of intersection on the predicted surface will be the corresponding point b as Figure 6 shows.

To calculate the energy between measured surface and predicted surface, Newcombe et al. uses point-to-plane algorithm as Figure 7shows.

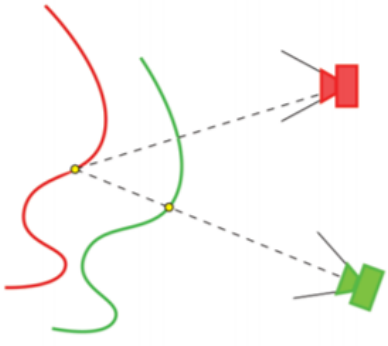

Figure 6: Finding corresponding point pairs. Red line denotes current measured surface, green line denotes predicted surface [6]

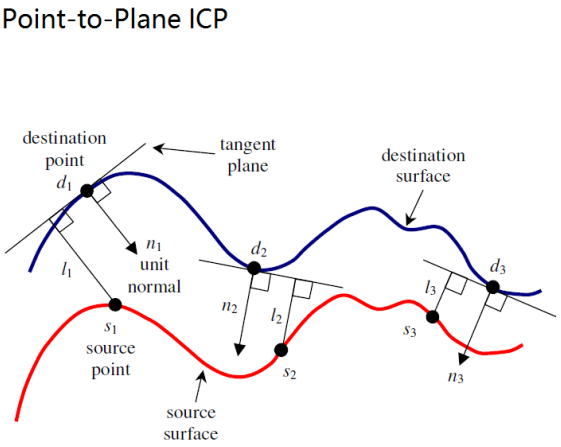

Figure 7: Point-to-plane ICP [3]

3 Experiments and Results

Newcombe et al. have conducted a series of experiments to test their system, Figure 8 and 9 shows some example output. The Kinect sensor is placed on a fixed turntable which will be spun through a full perfect circle resulting in $N = 560$ frames.

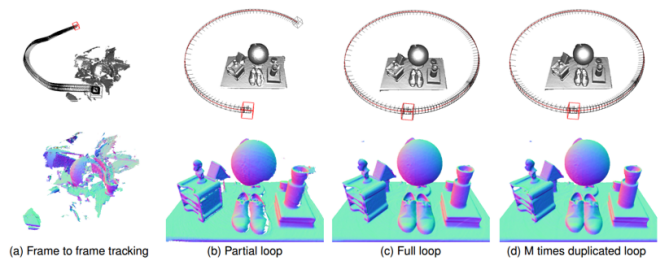

Figure 8: Experiments example output. The left one shows a comparison to this frame-to-model system, which used frame-to-frame tracking (560 frames were fed). The second one shows a result of partial loop (less than 560 frames were fed). The third one shows a result of one complete loop closure (560 frames were fed). The last one shows a result of four times duplicated loop closure (560 frames were fed repeatedly).[4]

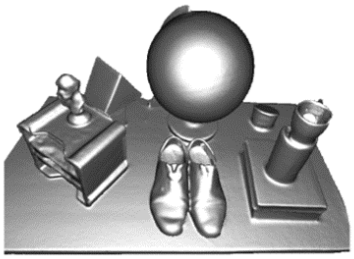

Figure 9: Sample result. 560×4 frames are fed from a free-moving Kinect sensor. Compared with previous set of experiments, we can see better result using frame-to-model algorithm. Besides, the result is remarkable for both perfect circle loop moving or free moving of the sensor.[4]

4 Discussion

According to the series of experiments, the performance of KinectFusion is remarkable with drift-free camera tracking and robustness. Even in darkness, the system could perform successfully. However, the main failure of the system occurs when reconstructing larger plane scene as there is few features which is hard to align.

5 Summary

KinectFusion is a real-time 3D reconstruction system introduced by Newcombe et al. in 2011. Rather than feature capture, the system fuse all data streamed from a Kinect sensor into a up-to-date global surface model. Using frame-to-model ICP algorithm to estimate camera pose. With the use of fully parallel algorithm, the system makes full use of commodity GPGPU hardware. However, challenges still remains. The system performs successfully to reconstruct indoor scene which is less than $7m^3$. For larger scene, too much memory will be needed leads to failure in real-time tracking and mapping. Besides, drift could occur as more pixels involved.

References

[1] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. 1996.

[2] Distributor. Using kinfu large scale to generate a textured mesh. `http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php`.

[3] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10), 2004.

[4] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, USA, 2011.

[5] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3dim*, volume 1, pages 145–152, 2001.

[6] Patrick Stotko. State of the art in real-time registration of rgb-d images. In *Central European Seminar on Computer Graphics for Students (CESCG 2016)*, 2016.