A Study on Surface Reconstruction from a Point Cloud Data Using Microsoft Kinect

Thesis
Submitted in partial fulfilment of the requirements of
BITS C421T/422T Thesis

By

SAI GOKUL K

2008B3A7375H

Under the supervision of

DR.TATHAGATA RAY

ASSISTANT PROFFESSOR,

COMPUTER SCIENCE,

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI HYDERABAD CAMPUS

30th April 2013

CERTIFICATE

This is to certify that the Thesis entitled, “A Study of Surface Reconstruction from a point cloud data using Microsoft Kinect” and submitted by **SAI GOKUL K** ID No. **2008B3A7375H** in partial fulfilment of the requirement of BITS C421T/422T Thesis embodies the work done by him/her under my supervision.

Signature of the Supervisor

Date: 30th April 2013
Dr.Tathagata Ray
Assistant Professor

ABSTRACT

The aim of the thesis was to understand, surface reconstruction from a point cloud data. This thesis focuses on how to effectively capture the input point cloud from the device Microsoft Kinect and has to be processed for the optimal results of the object reconstruction. Offline reconstruction algorithms like Cocone are applied to the point cloud data and also online reconstruction models have been studied and issues have been identified.

Contents

1. Introduction:

The purpose of this thesis is to enable a 3D object reconstruction from a point cloud. The point cloud is a data structure holding multi-dimensional data in it. Most commonly, point clouds are used to represent three-dimensional data and usually those are X, Y and Z coordinates.
In this thesis, it is described how point cloud data was sourced and processed for surface reconstruction. Online and offline algorithms were used and tested for surface reconstruction purpose. The need for the implementation comes from the point of view of a user who may want to create a 3D model but is not skilled in 3D modelling. The research goal of this thesis was to achieve better surface reconstruction using simpler and cheaper imaging devices like Microsoft Kinect rather than going for an expensive 3DScanner.

Generally 3D scanners are used to scan real world 3D objects to collect data on its shape and possibly its appearance (i.e. colour). The collected data can then be used to construct digital, three dimensional models. Collected 3D data is useful for a wide variety of applications. These devices are used extensively by the entertainment industry in the production of movies and video games[7].

The purpose of a 3D scanner is usually to create a point cloud of geometric samples on the surface of the subject. These points can then be used to extrapolate the shape of the subject If colour information is collected at each point, then the colours on the surface of the subject can also be determined.

3D scanners share several traits with cameras. Like cameras, they have a cone-like field of view, and like cameras, they can only collect information about surfaces that are not obscured. While a camera collects colour information about surfaces within its field of view, a 3D scanner collects distance information about surfaces within its field of view. The "picture" produced by a 3D scanner describes the distance to a surface at each point in the picture. This allows the three dimensional position of each point in the picture to be identified.

For most situations, a single scan will not produce a complete model of the subject. Multiple scans, even hundreds, from many different directions are usually required to

obtain information about all sides of the subject. These scans have to be brought in a common reference system, a process that is usually called *alignment* or *registration*, and then merged to create a complete model. [7]

The point clouds produced by 3D scanners can be used directly for measurement and visualization in the architecture and construction world.

2. USED TOOLS AND TECHNOLOGIES

2.1 Microsoft Kinect:

Microsoft's Kinect as shown in Figure 1 is described as a "controller-free gaming and entertainment experience" and is commonly sold bundled with the Xbox 360. But to see it as just another way to play games is to underestimate its significance. It is a general purpose and low-cost 3D input device not to mention its amazing audio input capabilities.

Fig.1 Microsoft Kinect

Essentially this hardware has cameras that makes use of infra-red (IR) illumination to obtain depth data, color images and sound. The IR is used as a distance ranging device.

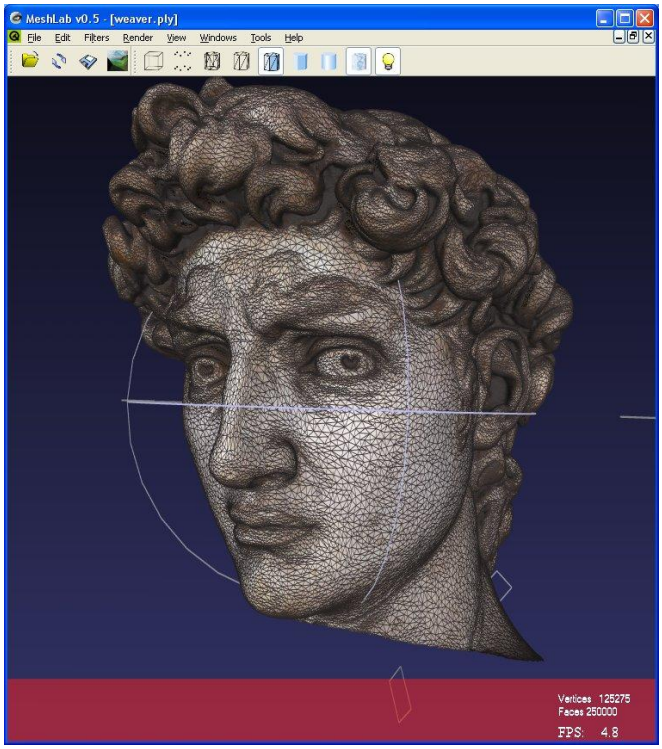
A custom chip processes the data to provide a depth field that is correlated with the color image. That is the software can match each pixel with its approximate depth. The preprocessed data is fed to the machine via a USB interface in the form of a depth field map and a color image.[4]

2.2 Kinect SDK

The Microsoft Kinect for Windows SDK provides the native and managed APIs and the tools needed to develop Kinect for Windows applications.[6]

2.3 Meshlab

MeshLab(Figure.2) is an open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes. The system is aimed to help the processing of the typical not-so-small unstructured models arising in 3D scanning, providing a set of tools for editing, cleaning, healing, inspecting, rendering and converting this kind of meshes.[8]



Firgure 2 Screenshot of Meshlab with a point cloud data file

2.4 Geomview

Geomview(Figure.3) is an interactive 3D viewing program for Unix. *Geomview* lets you view and manipulate three-dimensional objects: you use the mouse to rotate, translate, zoom in and out, and so on. *Geomview* can be used as a standalone viewer for static objects or as a display engine for other programs which produce dynamically changing

geometry. *Geomview* can display objects described in a variety of file formats.[9]

Figure 3. Screenshot of Geomview

3. RESEARCH FUNDAMENTALS AND ISSUES:

3.1. Surface Reconstruction:

Reconstructing surfaces from sample points in a faithful way is a difficult challenge. The problem appears in applications such as CAD designs, medical imaging, visualization, reverse engineering and so on. Recent advances in modern laser technology have made it easier to generate a lot of sample from the boundary of an object. The reconstructed surface should be topologically equivalent to the sampled surface and also geometrically close. For point clouds we initially used an offline algorithm developed by Dr.T.K Dey et al. [1] from Ohio State University which is a Voronoi based algorithm called *Cocone(complimented cones)* [2]. This uses a single *Voronoi* diagram computation. For dense samples from smooth surfaces this algorithm works nicely. For surfaces with boundaries, for water-tight surfaces, for large data sets and noisy point clouds they have developed different versions of Cocone which are called *Cocone, Tight Cocone, Super Cocone and Robust Cocone.*

Also there is an algorithm developed called *AMLS (Adaptive Moving Least Squares) method* [3] for smoothing noisy point cloud data before applying cocone.

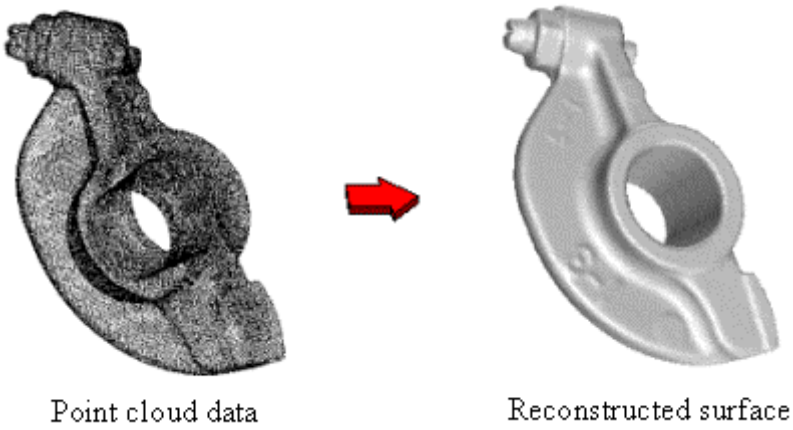


Figure 4. Sample reconstruction of a surface from a point cloud

Later on as we progressed, online algorithms were applied for surface reconstruction which primarily aimed at real time surface reconstruction.

3.2 Point Cloud data

A point cloud is a data structure holding multi-dimensional data in it. Most commonly, the point clouds are used to represent the three-dimensional data and usually those are X, Y and Z coordinates. The common usage for the point data is to represent an outer surface of some object. Beside the geometrical coordinates, the point clouds can contain colour data for each point, normal directions or both. When data is added to a point, the cloud dimensions increase. When the colour data is added to the three-dimensional geometric cloud, it becomes four-dimensional.

Industrial quality equipment is not mandatory anymore for the object scanning, since with the current consumer class equipment, such as Microsoft Kinect or Asus XTionPRO, it is possible to acquire point clouds with a decent accuracy.

3.3 Reconstruction pipeline

There are a set of steps that were followed in order for the offline reconstruction of the surface from the point cloud data which can be visualized from the flowchart below:

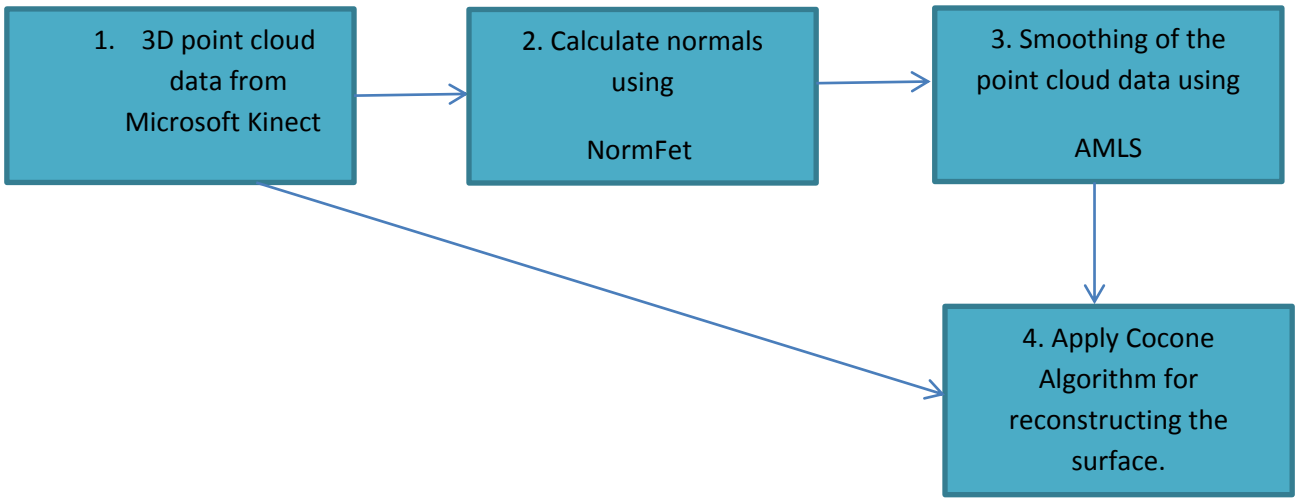


Figure 5. Offline Surface reconstruction pipeline

Step-1:

The Kinect SDK offers an API through which we can capture the data.

`DepthImageFrame` class is used for depth images. It provides access to dimensions, format and pixel data for a depth frame.

The individual pixel information is extracted keeping a static bounding box as a reference.

These pixels are outputted to an .off file consisting of n points and 3n coordinates which looks like the Figure 6 below when viewed in MeshLab.

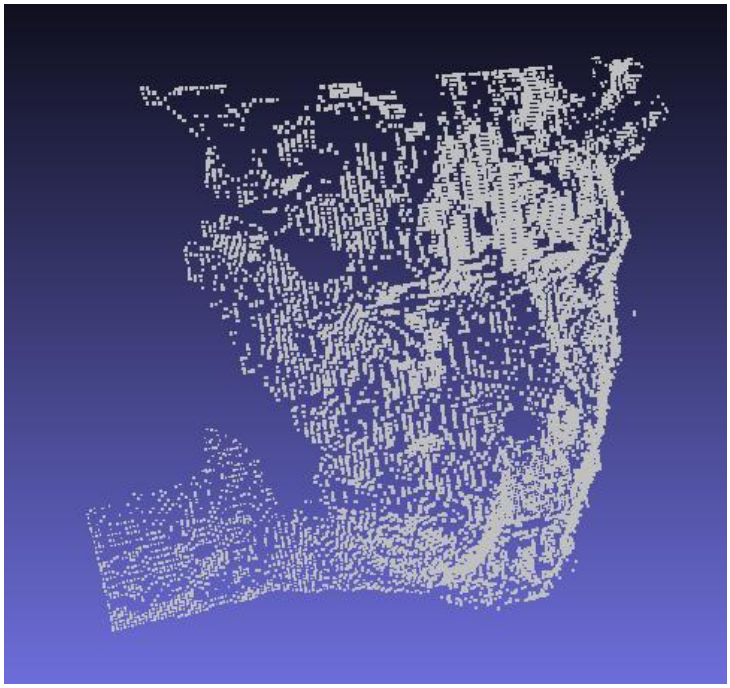


Fig.6. Point cloud data with approximately 7000 points.

Step-2:

The normals are calculated using the NormFet software. The normal are needed for the AMLS method to work. It isn't mandatory for the Cocone algorithm.

Step-3:

After the computation of the normals, AMLS is applied which results in an n point 3n coordinate OFF file.

Step-4:

When the Cocone is run on the smoothened OFF file, it reconstructs the surface as can be seen below in figure 7,

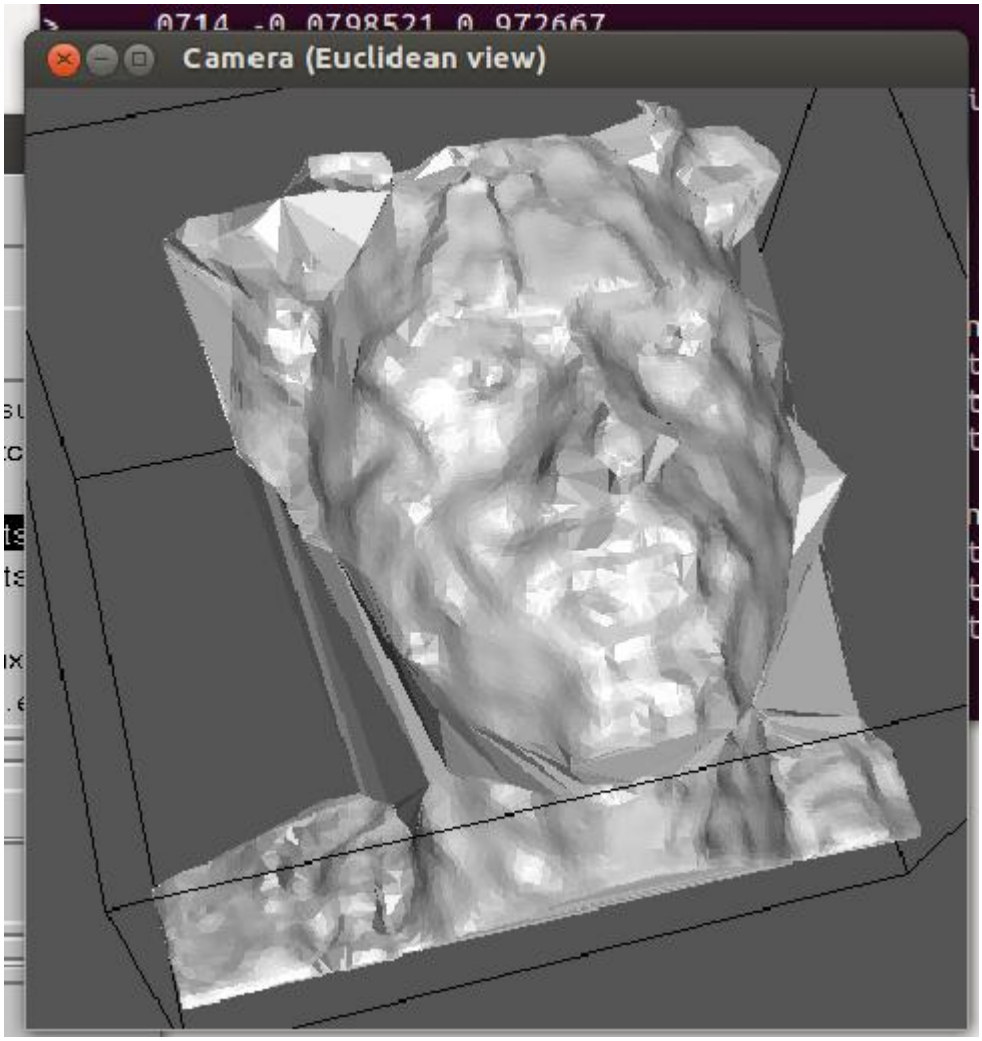


Figure 7. Reconstructed surface from processed point cloud data

The surface reconstruction can also be done without smoothing out the data, whose output is shown below in figure 8.

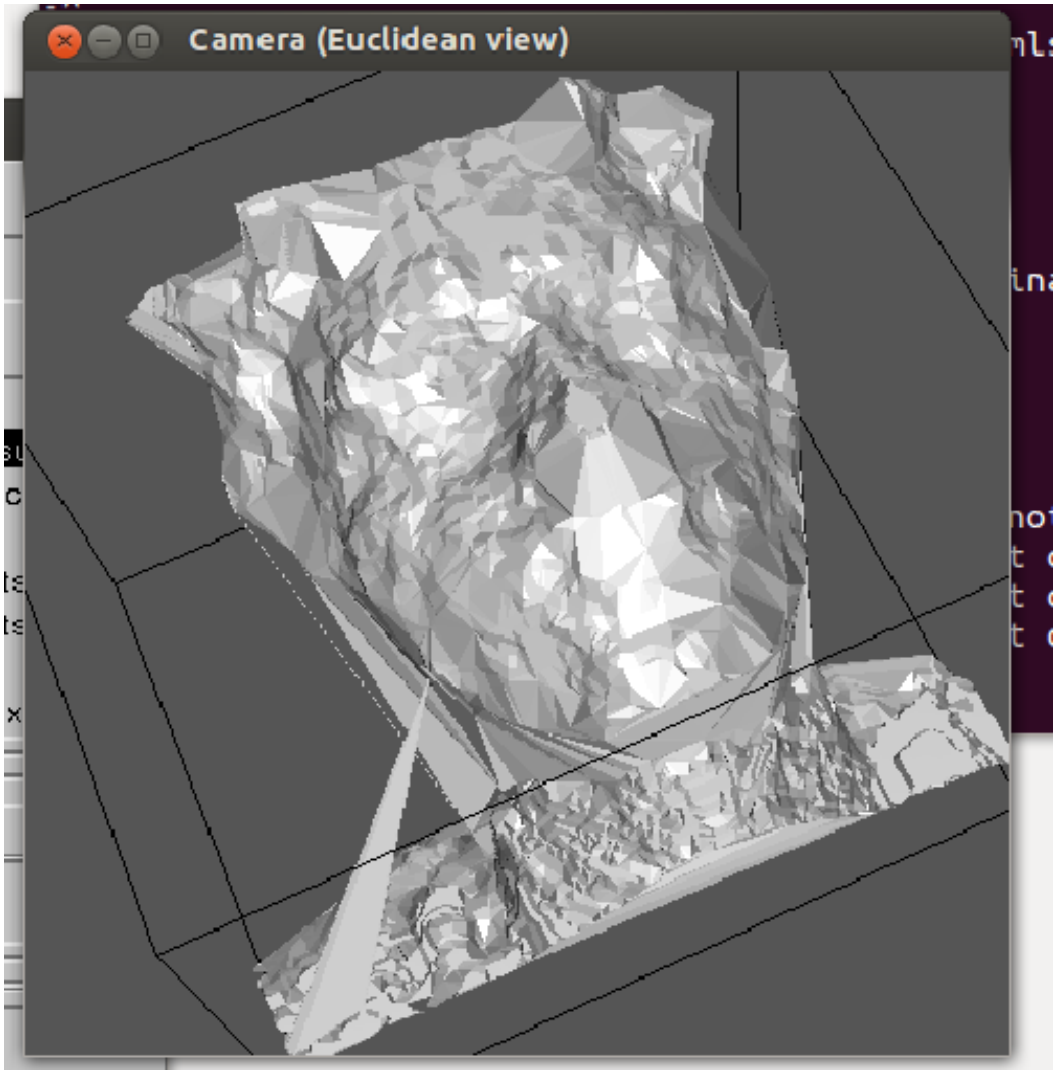


Fig.8 . Reconstructed surface from raw point cloud data

As we can clearly observe, the output from the smoothened data results in a more realistic surface rather than the output from the original data which has extrapolations and sharp attributes. Also on closer observation, some input point cloud data resulted in holes which is not desirable.

- ➔ To improve the reconstructed surface quality, we need to improve the point cloud data resolution.
- ➔ In order to do that, a method has been proposed so as to scan an object from multiple angles and merge into one thus creating much richer and denser data.

Following two methods have been tried in order to achieve a better point cloud.

1. ReconstructMe
2. Kinect Fusion

These will be described in the forthcoming sections.

4. ReconstructMe

ReconstructMe[10] is a tool that can be used to perform multiple scans and merge them into a single point cloud. It is capable of doing both offline and online processing. It also supports scanning from multiple devices.

Offline processing implies that the scans are done first and later gets converted to point cloud whereas online processing implies that the scan and generation of point cloud are done simultaneously.

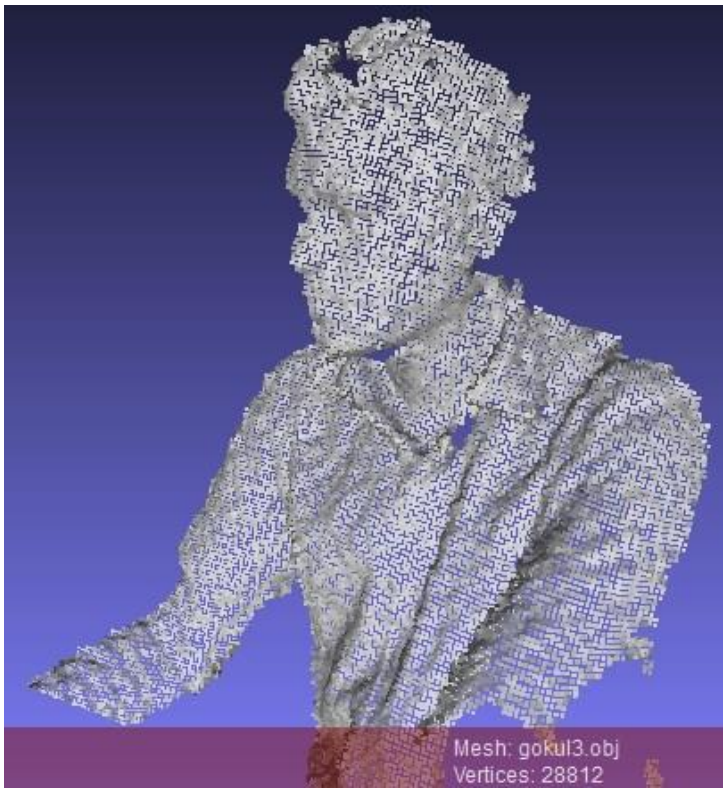


Fig.9 Point cloud data with approximately 28000 points

As you can see from figure 9, the point cloud density has increased from 7000 points (fig.6) to 28000 points.

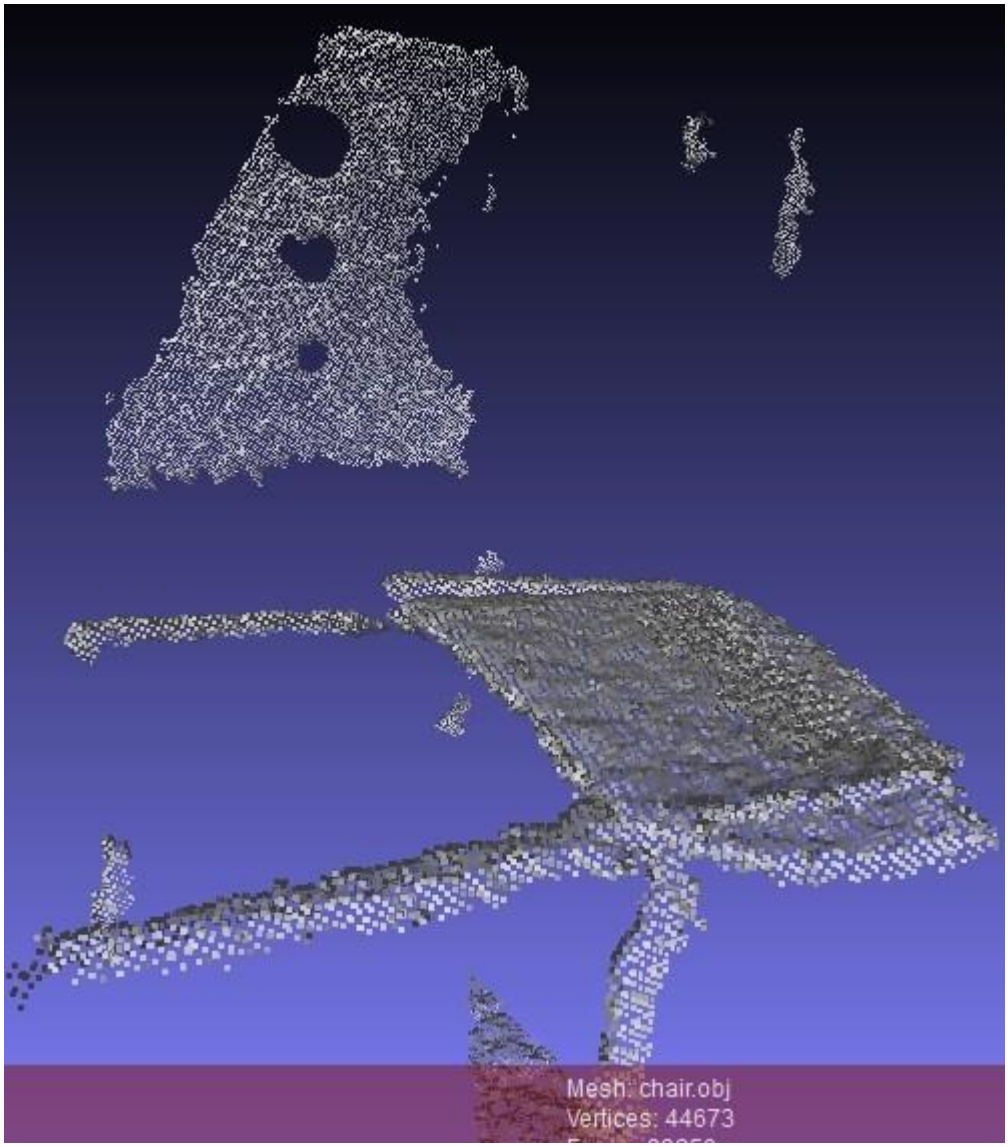


Fig.10 . The disoriented chair point cloud data

In the above figure 10, the chair can be seen disoriented.

4.1. Issues:

- ➔ The resulting point cloud density is better than the previous attempt, but there are holes present.
- ➔ The computational ability of the test PC is not sufficient to handle the frame rate while scanning.

  The test PC has the following configuration:
  - AMD Dual-Core E-350 Accelerated Processor 1.6GHz
  - 6GB DDR3 SDRAM

- • AMD Radeon™ HD 6310 Discrete-Class Graphics with ~3GB shared memory and DirectX 11 support.

➔ The reconstruction volume gets reset due to the same reason and hence 360 degree scanning is not possible.

➔ Even the tiniest of the movement of the scanner produces new frames to be scanned and added to the point cloud.

➔ Hence the resulting reconstruction would end up being inaccurate.
Example: overlapping of surfaces, disorientation, partial reconstruction

➔ Cocone algorithm gives a wrong reconstruction for the point cloud generated by this method. It simply gives the convex hull of the point cloud(reconstruction volume) as shown in figure 11.

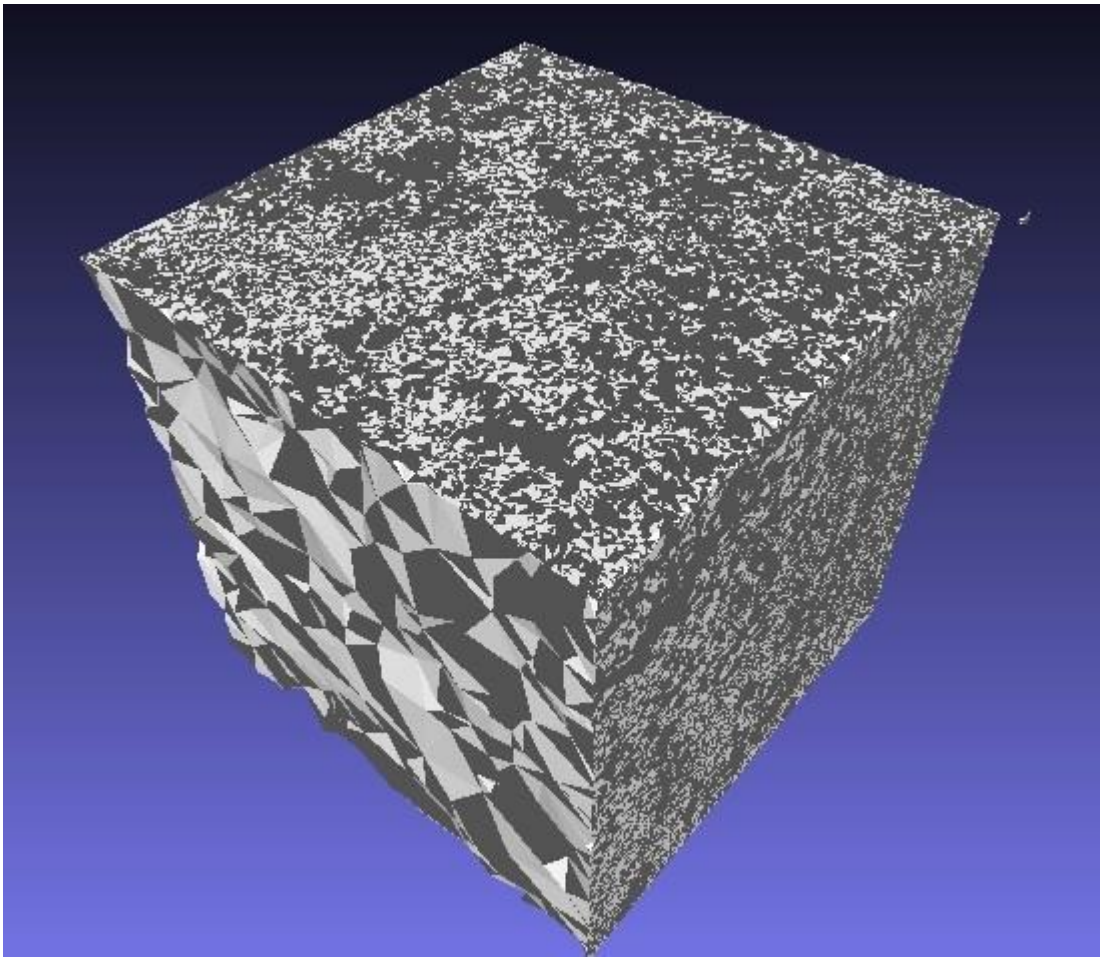


Fig.11 Wrong reconstructed surface .Bounding box of the point cloud of Fig.9

5. Kinect Fusion

Kinect Fusion[11] is an algorithm developed by Microsoft Research in 2011. The algorithm allows a user to reconstruct a 3D scene in real-time and robustly by moving the Microsoft Kinect sensor around the real scene. Kinect Fusion's results possess both a high degree of robustness and detail.

Kinect Fusion requires a Microsoft Kinect sensor. The sensor is equipped with a RGB camera (640x480 pixels at 30 FPS), an infrared projector and an infrared sensor that, combined, provide a depth map of the scene (640x480 pixels at 30 FPS), a four-microphone array with built-in noise suppression and directionality estimation and a tilt motor .Internally, the Kinect has a processor for data acquisition, manipulation and transmission.

Other than the Kinect, for Kinect Fusion to work a PC equipped with a CUDA-enabled Nvidia GPU is required, used for fast parallel processing. The processor is also required to be of high power, as the approximations done by Kinect Fusion are acceptable only in the case the algorithm can run at least at 30 frames per second.

Kinect Fusion has advantages compared to ReconstructMe :

- ➔ Produces denser point cloud.
- ➔ Has more control on depth range.
- ➔ Performs real time reconstruction better than ReconstructMe.
- ➔ Reconstruction volume voxels can also be controlled (max being 640x640x640).

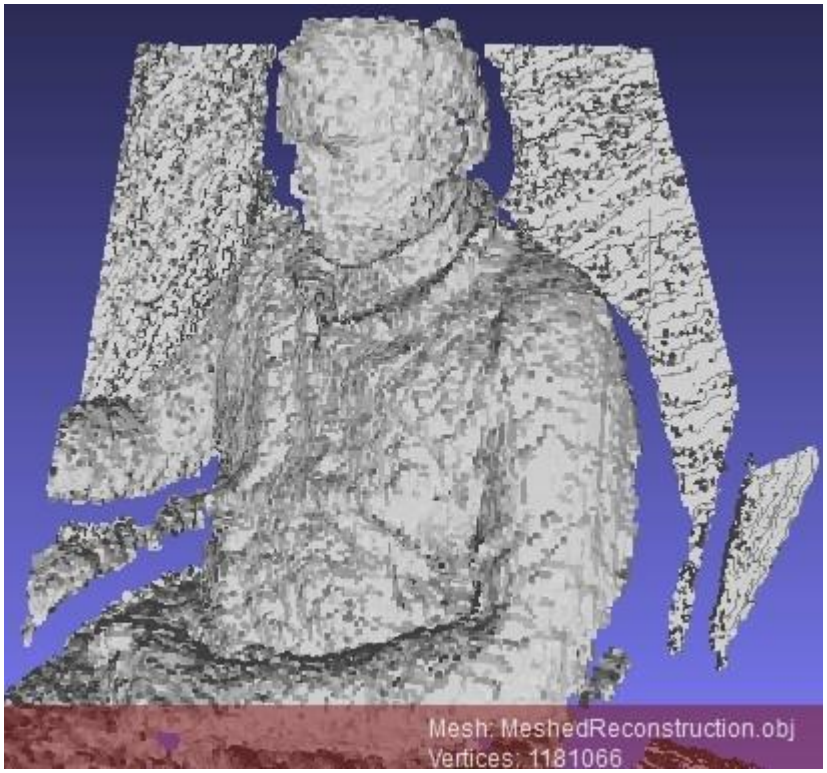


Fig.12 Point cloud data with number of points close to 1 million

As you can see in figure 12, the total points scanned are almost close to 1 million points.

5.1. Issues:

- ➔ Desktop PC with 3GHz (or better) multi-core processor and a graphics card with 2GB or more of dedicated on-board memory.
- ➔ The minimum hardware requirement for GPU based reconstruction is a DirectX 11 compatible graphics card. Kinect Fusion WILL NOT RUN on hardware that does not meet this requirement.
- ➔ Though the point cloud density is high, 360 degree scan is still not possible due to the computational difficulties.
- ➔ Also the reconstructed surface has contours that are out of alignment, possibly due to poor Iterative Closest Point Algorithm(ICP)[17] output which is indirectly related to the handling of input frame rate.

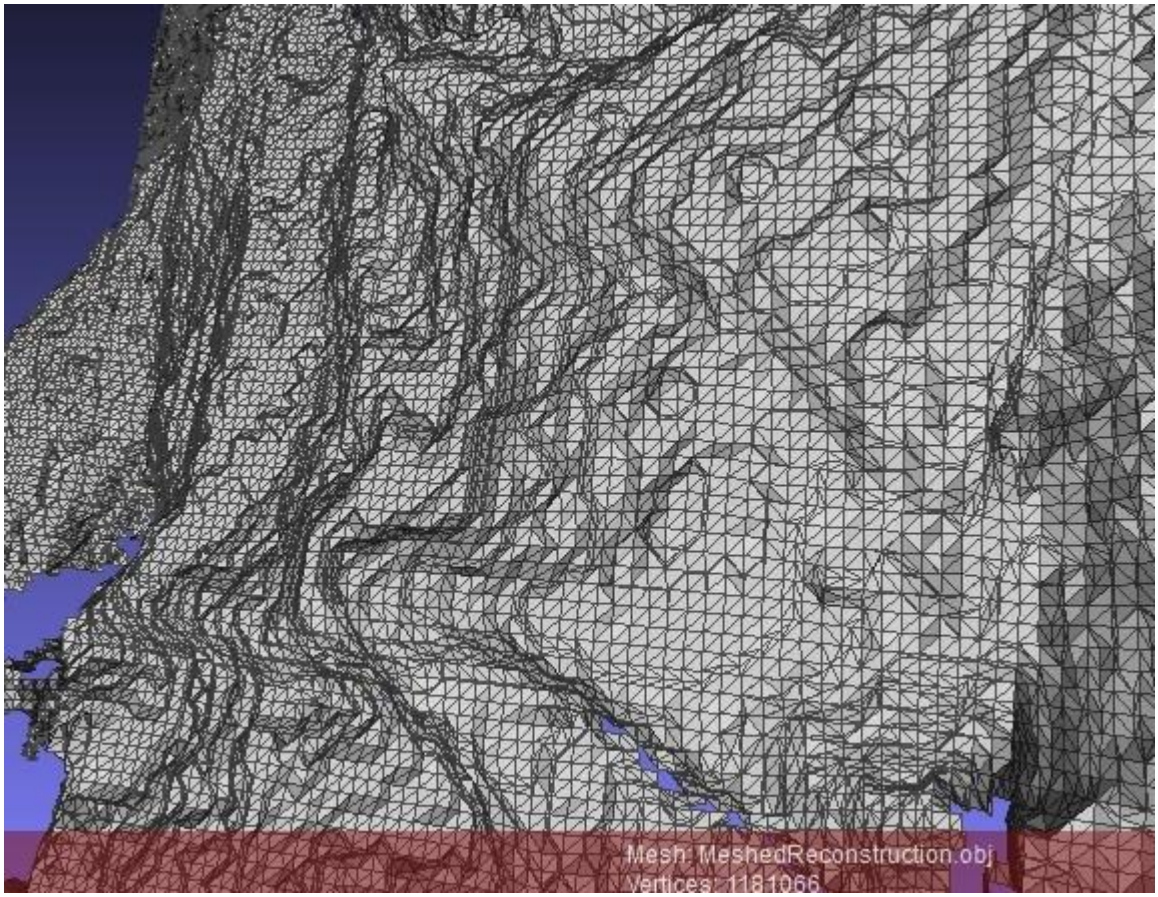


Fig.13 Contours out of alignment in the reconstructed surface

The Contours can be seen out of alignment in the figure 13.

The following sections describe the functioning of the Kinect Fusion.

6. Kinect Fusion – How it functions

The Kinect Fusion system reconstructs a single dense surface model with smooth surfaces by integrating the depth data from Kinect over time from multiple viewpoints. The camera pose is tracked as the sensor is moved (its location and orientation) and because we now know each frame's pose and how it relates to the others, these multiple viewpoints of the objects or environment can be fused (averaged) together into a single reconstruction voxel volume. A large virtual cube in space (the reconstruction volume), located around the scene in the real world, and depth data (i.e. measurements of where the surfaces are) are put (integrated) into this as the sensor is moved around.

The input of the Kinect Fusion algorithm is a temporal sequence of depth maps returned by the Kinect. Since the algorithm is using just the depth maps and no colour information, lighting conditions do not interfere with it, allowing Kinect Fusion to function even in complete darkness. The algorithm runs in real-time, so it proceeds by using one input depth frame after the other as it is fed from the sensor. A surface representation is extracted from the current depth frame and a global model is refined by first aligning and then merging the new surface with it. The global model is obtained as a prediction of the global surface that is being reconstructed and refined at each new step.[12][13]

6.1. Three stages of the algorithm:

- The **first stage** is depth map conversion. At each new frame, a new depth map obtained from the Kinect sensor is used as input.

  A depth map is a pixel image that, instead of holding color values, holds depth values, i.e. distances from the camera to the 3D scene points. Kinect computes the depth map by emitting a non-uniform infrared pattern on the scene through its infrared projector and then acquiring the same pattern through an infrared sensor, measuring its time-of-flight (TOF)[14]. Holding hard-coded images of the light pattern, the processor inside the Kinect is able to match the incoming pattern to the hard-coded images, it finds correlations and thus computes the 3D positions of the points in the scene, building the depth map[15].

In order for the Kinect Fusion algorithm to work, the depth map must be converted into a 3D point cloud with vertex and normal information. This is done at different, layered resolutions, resulting in a number of images with different levels of detail.

The depth map is converted into a 3D point cloud through back-projection. To do so, the Kinect's internally stored calibration matrix is used, that matches depth pixels to actual 3D coordinates. The normal of each point is estimated through cross-product of two vectors: the vector joining the chosen point and the one above it, and the vector joining the chosen point and the one to its right. Since the points are arranged as the depth pixels, choosing point (x,y,z) the points (x,y+1,z) and (x+1,y,z) are taken for normal estimation, with x and y being the image coordinates and z being the depth at those coordinates.

The result of this step is a point cloud with vertex and normal data for each point at three different levels of detail. Note that the cloud is considered ordered, as the points are arranged as the depth map's pixels.

- The **second** stage calculates the global/world camera pose (its location and orientation) and tracks this pose as the sensor moves in each frame using an iterative alignment algorithm, so the system always knows the current sensor pose relative to the initial starting frame. To perform alignment, the Iterative Closest Point algorithm (ICP)[16,17,18] is used.

  The ICP algorithm performs alignment of two point clouds, called source and target as shown in fig.14. After ICP executes successfully, the source will be aligned to the target. The algorithm requires the source cloud to be already close to the correct match, as such ICP is used for refinement.

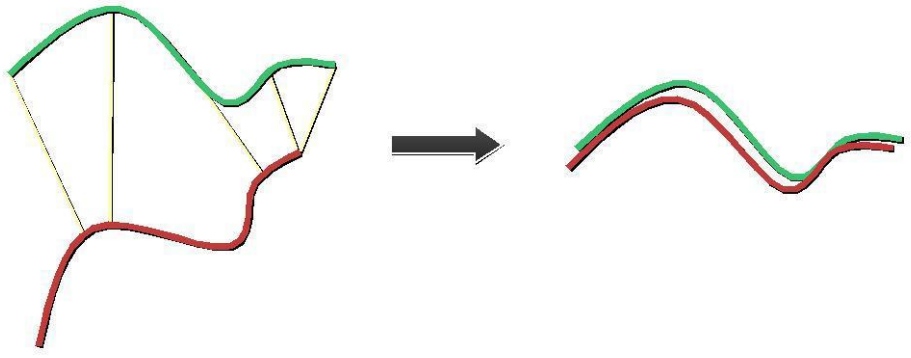


Fig.14. How an ICP step works in 2D

Kinect Fusion does not use the standard ICP algorithm, since it is too slow for its real-time purposes. Instead, it makes the assumption that the changes between the current cloud and the previous one are small. The assumption is reasonable if the algorithm runs in real-time (at 30 frames per second) and if the Kinect is moved around the scene at a slow pace. If the processor, however, is not able to cope with the speed, the ICP algorithm will fail and the new cloud will be discarded. The modified ICP then back-projects the two clouds onto the camera image frame of the model and considers two points to be a match if they fall on the same pixel. This allows the algorithm to be parallelized on GPU. To further speed up the computation, the ICP iterations are performed at the three resolutions, starting from the coarser one, and the transformation matrix is computed in the GPU by taking advantage of the incremental nature of the transform (due to the small difference between two consecutive frames). Thanks to this, the modified ICP algorithm presents a complexity of only O(S), with S is the number of iteration steps, which is kept small by the use of the multi-resolution pyramid. In addition, the assumption of small changes between two subsequent frames allows ICP to be used without resorting to another algorithm for initial alignment.

After a match is found, the modified ICP algorithm computes the error between the two points in a match with a point-to-plane metric, instead of a point-to-point metric. This means that the error is computed as the distance between the plane parallel to the surface at the first cloud point and the position of the second cloud point. This metric has been shown to converge faster than the point-to-point metric[19].

ICP, after a set of iterations, generates a six degrees of freedom transformation matrix that aligns the source point cloud to the target point cloud with a rotation and a translation.

- The **third** stage is fusing (or “integration”) of the depth data from the known sensor pose into a single volumetric representation of the space around the camera. Once the alignment transform is found, and thus the current pose of the camera is estimated, the new cloud can be merged with the current model. The raw depth data is used for merging instead of the filtered cloud to avoid losing details. A Truncated Signed Distance Function (TSDF) is used for

this step. This function basically extracts the surface of the objects in the scene and assigns negative numbers to those pixels that are either inside objects or inside an area we have not yet measured, positive numbers to pixels that are outside the surface, increasing the further they are from it, and zero to the pixels that are on the surface .A TSDF is computed for the new cloud and merged to the current model's TSDF through a weighted running average to compute the new surface.

This representation has been chosen by the authors mainly due to the ease of merging different TSDFs just by weight averaging and due to the ease of parallelization, which allowed the authors to achieve real-time performance.

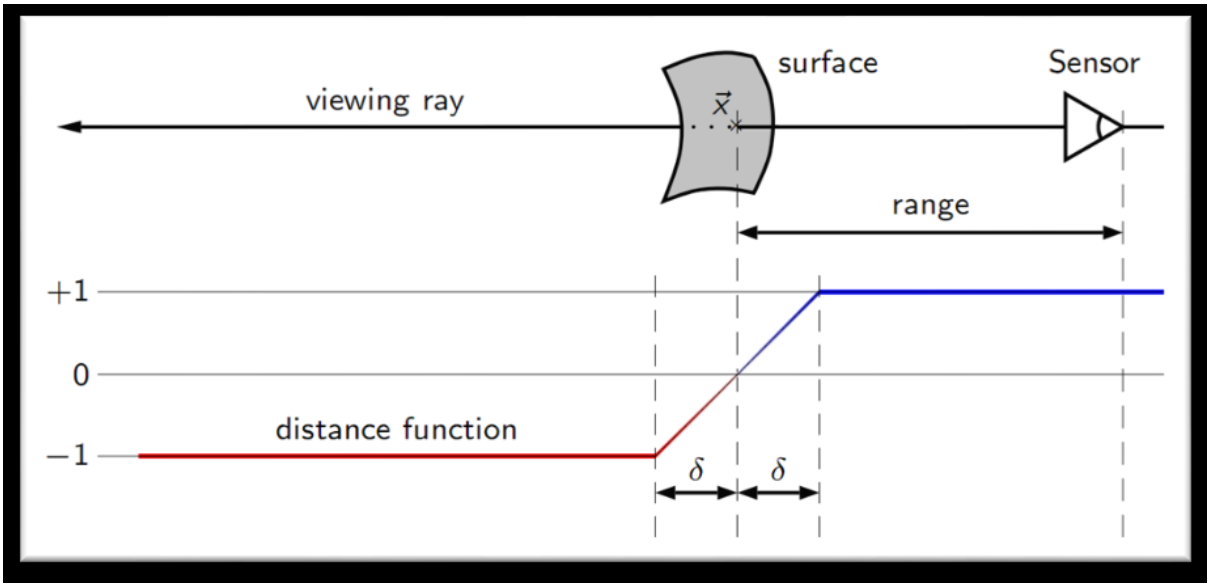


Fig.15. How a TSDF is created

The resulting TSDF as shown in figure 15 is used to reconstruct the surface of the current, refined model. To do so, ray casting is performed.

- The reconstruction volume can be raycast [20] from a sensor pose (which is typically, but not limited to, the current Kinect sensor pose), and this resultant point cloud can be shaded for a rendered visible image of the 3D reconstruction volume. In Kinect Fusion ray casting is used for the surface reconstruction step. Ray casting is performed from the global camera focal point by intersecting the zero level set of the TSDF. A prediction of the current global surface is obtained, with vertex data and estimated normal data. The refined, ray-casted model is the one that will be used in the next ICP step for

alignment. By doing so instead of using just the last frame point cloud as the source for alignment, a less noisy model is obtained, placing the focus on finding a smooth surface. The process is sped up by using ray skipping, that is by advancing the rays at discrete steps. Ray casting is also used for the final visualization step. The end result is a 3D surface representing the acquired scene.

6.2. Constraints on Tracking

Kinect Fusion depends on depth variation in the scene to perform its camera tracking. Scenes must have sufficient depth variation in view (and in the model) to be able to track successfully. Small, slow movement in both translation and rotation are best for maintaining stable tracking. Dropped frames can adversely affect tracking, as a dropped frame can effectively lead to twice the translational and rotational movement between processed frames.

6.3. Reconstruction Volume

The reconstruction volume is made up of small cubes in space, which are generally referred to as Voxels. The number of voxels that can be created depends on the amount of memory available to be allocated on your reconstruction device, and typically up to around 640x640x640 = 262144000 voxels can be created in total on devices with 1.5GB of memory or more.

On GPUs the maximum contiguous memory block that can typically be allocated is around 1GB, which limits the reconstruction resolution to approximately 640^3 (262144000 voxels). Similarly, although CPUs typically have more total memory available than a GPU, heap memory fragmentation may prevent very large GB-sized contiguous memory block allocations.. If you need very high resolution also with a large real world volume size, multiple volumes or multiple devices may be a possible solution.

7. Online Triangulation Algorithm

This method computes a triangulation of the point stream generated by the laser scanner online, i.e., the data points are added to the triangulation as they are received from the scanner. Multiple scanned areas and areas with a higher point density result in a finer mesh and a higher accuracy. On the other hand, the vertex density adapts to the estimated surface curvature. To assist the human operator the resulting triangulation is rendered with a visualization of its faithfulness. Additionally, this triangulation method allows for a level-of-detail representation to reduce the mesh complexity for fast rendering on low-cost graphics hardware.[21]

This method primarily aims to address the following issues:

A. Handling of large point sets,

B. Handling of heterogeneous point densities of incoherently scanned regions,

C. Handling of high precision point data with predefined measurement errors,

D. Handling of point streams of arbitrary order, i.e., online triangulation,

E. Triangulating full-automatically,

F. Triangulating in real-time, and

G. Assisting the human operator of the scanner.

The algorithm follows a series of steps:

ONLINE-TRIANGULATION(d_1,d_2, . . .)
Input: Data point stream D = (d_1,d_2, . . .);
Output: Triangulation T approximating {p_1, p_2, . . .}.
while (D not terminated) do {
ADDTONEIGHBORHOODBALLS(d_i);
Update normals of affected neighborhood balls;
Update local approximation;
Triangulate area of affected neighborhood balls;
Render triangulation with uncertainty visualization;
}

This algorithm hasn't been implemented on Microsoft Kinect.

8. Conclusion/Further course of action:

➔ The following table summarizes the performance of the surface reconstruction against respective graphic cards.[22]

Chart Legend
Smooth realtime experience
Jerky realtime experience
Does not work at all

Model	Standard	Highres
ATI FirePro V3800	3	1
ATI FirePro V4800	3	1
ATI FirePro V5900	3	3
ATI Radeon HD 2600 Pro	2	1
ATI Radeon HD 4570M	1	1
ATI Radeon HD 4870	1	1
ATI Radeon HD 5145M	1	1
ATI Radeon HD 5650M	3	2
ATI Radeon HD 5700	3	3
ATI Radeon HD 6490M	2	1
ATI Radeon HD 6570	3	2
ATI Radeon HD 6850	3	3
ATI Radeon HD 7640G	2	1
ATI Radeon HD 7850	3	3
INTEL Dual Quadcore	2	2
INTEL HD Graphics 3000	2	1
NVIDIA Geforce 9500GT	2	1
NVIDIA Geforce 9800GT 3D	3	3
NVIDIA Geforce 9800GTX	3	3
NVIDIA Geforce G 102M	2	1
NVIDIA Geforce GT 240	3	2
NVIDIA Geforce GT 330M	2	1
NVIDIA Geforce GT 430	3	3
NVIDIA Geforce GT 440	3	3
NVIDIA Geforce GT 555M	3	3
NVIDIA Geforce GT 630M	3	3
NVIDIA Geforce GTX 220	2	2
NVIDIA Geforce GTX 280	3	2
NVIDIA Geforce GTX 295	3	1
NVIDIA Geforce GTX 460	3	3
NVIDIA Geforce GTX 550Ti	3	3
NVIDIA Geforce GTX 560	3	3
NVIDIA Geforce GTX 560M	3	3
NVIDIA Geforce GTX 570	3	3
NVIDIA Geforce GTX 590	3	3
NVIDIA Geforce GTX 660M	3	3
NVIDIA Geforce GTX 670M	3	3

NVIDIA Geforce GTX 680	3	3
NVIDIA GT 540M	3	3
NVIDIA Quadro 2000	2	2
NVIDIA Quadro 3000M	2	2
NVIDIA Quadro 4000	3	2
NVIDIA Quadro FX 2800M	3	1
NVIDIA Quadro FX 3500M	3	1
NVIDIA Quadro FX 4800	3	3
NVIDIA Quadro FX 580	3	1
NVIDIA Quadro NVS 140M	1	1
NVIDIA Quadro NVS 295	2	1
NVIDIA Quadro NVS 3100M	2	1
NVIDIA Quadro NVS 4200M	2	1

- ➔ The algorithm mentioned in section 7 shall be implemented using Kinect as the scanner.
- ➔ Also the Kinect Fusion shall be tried on a PC with the following minimum configuration :
  - Desktop PC with 3GHz (or better) multi-core processor
  - NVidia GeForce GTX560/ AMD Radeon 6950 and above with 2GB on board Graphics.

and resulting reconstruction shall be studied and compared.

References

1. http://www.cse.ohio-state.edu/~tamaldey/surfrecon.htm
2. *N. Amenta, S. Choi, T. K. Dey and N. Leekha.* A simple algorithm for homeomorphic surface reconstruction. *Proc. 16th Sympos. Comput. Geom., 2000, 213--222.*
3. *T. K. Dey and J. Sun*. An Adaptive MLS Surface for Reconstruction with Guarantees.*IEEE Symposium on Geometry Processing, (2005), 43--52.*
4. http://www.i-programmer.info/ebooks/practical-windows-kinect-in-c/3725-getting-started-with-windows-kinect-sdk-10.html
5. https://publications.theseus.fi/
6. http://www.microsoft.com/en-us/kinectforwindows/
7. http://en.wikipedia.org/wiki/3D_scanner
8. http://meshlab.sourceforge.net/
9. http://www.geomview.org/
10. http://reconstructme.net/
11. *KinectFusion: Real-Time Dense Surface Mapping and Tracking* Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon - October 2011 - IEEE ISMAR
12. http://msdn.microsoft.com/en-us/library/dn188670.aspx
13. *KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera* Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon - October 2011 - ACM Symposium on User Interface Software and Technology.
14. http://www.wired.com/gadgetlab/2010/11/tonights-release-xbox-kinect-how-does-it-work/
15. http://courses.engr.illinois.edu/cs498dh/fa2011/lectures/Lecture%2025%20-%20How%20the%20Kinect%20Works%20-%20CP%20Fall%202011.pdf
16. P. Besl and N. McKay. A method for registration of 3D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 14(2):239–256, 1992
17. http://en.wikipedia.org/wiki/Iterative_closest_point
18. http://pointclouds.org/documentation/tutorials/iterative_closest_point.php
19. http://www.comp.nus.edu.sg/~lowkl/publications/lowk_point-to-plane_icp_techrep.pdf
20. http://en.wikipedia.org/wiki/Ray_casting
21. *Online Triangulation of Laser-scan Data*, Klaus Denker, Burkhard Lehner, and Georg Umlauf Geometric Algorithms Group, University of Kaiserslautern, Germany
22. https://docs.google.com/spreadsheet/ccc?key=0AjYhEvwkxrJOdHBGaTMyWVVBNVFjRHFzbU5RQU81TWc