OULUN SEUDUN
AMMATTIKORKEAKOULU

Miika Santala

3D CONTENT CAPTURING AND RECONSTRUCTION USING MICROSOFT KINECT DEPTH CAMERA

3D CONTENT CAPTURING AND RECONSTRUCTION USING MICROSOFT KINECT DEPTH CAMERA

Miika Santala
Bachelor's thesis
Spring 2012
Degree Programme in Information Technology and Telecommunications
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology and Telecommunications

---

Author: Miika Santala
Title of thesis: 3D Content Capturing and Reconstruction Using Microsoft Kinect Depth Camera
Supervisors: Pekka Alaluukas, Jarkko Vatjus-Anttila
Term and year when the thesis was submitted: Spring 2012 Pages: 39 + 3 appendices

---

The main motivation behind this thesis was to create a new method of interaction between the physical and virtual space.

The object was set to be achieved by creating a 3D-scanner capable of producing 360-degrees scans of real world objects to be visualized and used inside virtual spaces. The implementation of this objective began by conducting a background research about the existing methods for acquiring depth data of the physical objects and about integrating multiple scans to form a solid 360-degree presentation.

As a result, the implementation of this work is able to obtain depth scans of the objects positioned on planar surfaces, filter the background data out from it and merge multiple scans in to a single point cloud presentation of the original object.

Further methods of processing the data for the purposes of real time rendering are addressed in the bachelor thesis of Juha Hyvärinen.

Keywords:
cameras, virtual reality, image reconstruction, computer vision

TABLE OF CONTENTS

SYMBOLS AND ABBREVATIONS

nD	n-Dimensional
RGB	Red, green and blue components
SLR	Single-lens reflex
HDR	High Definition Range
RGB-D	Red, green, blue and depth components
SIFT	Scale Invariant Feature Transform
ICP	Iterative Closest Point
RANSAC	Random Sample Consensus
SDK	Software Development Kit
API	Application Programming Interface
IR	Infra-Red
NNS	Nearest Neighbor Search
PCL	Point Cloud Library
SVD	Singular Value Decomposition
RAM	Random Access Memory
Point cloud	Collection of n-Dimensional points

1 INTRODUCTION

This thesis was written for the Chiru project at University of Oulu, Center for Internet Excellence (CIE), Intel and Nokia Joint Innovation Center. Chiru is a three year long project started by Intel, Nokia and CIE, aimed to research the 3D-internet and new user interface paradigms.

The motivation for this thesis was to implement a new method for interaction between the physical and virtual spaces, by creating a processing chain capable of capturing the object geometry and texture of the physical real world objects.

The thesis is structured as follows: first, an overview of the previous work in the field of 3D reconstruction is given. Then, a brief introduction for the tools and technologies used to accomplish the goal of the thesis are provided. A great deal of terminology is involved with the 3D reconstruction, which is explained to help the user to follow the text.

From the basics, the thesis continues to explain the implementation of the 3D reconstruction processing chain. The performance and the capabilities of the implementation are measured and presented before touching the subject of integrating the implementation to realXtend Tundra virtual world framework.

2 RESEARCH

The problem of capturing and reconstructing real world objects for 3D visualization can be divided in two key areas: how to obtain depth data of the real world objects, and how to merge multiple captures of the object into a single virtual presentation. Both of these problems are extensively discussed in the research of the field.

2.1 Data acquisition

The technologies for obtaining depth data can be generally divided in two categories: contact and non-contact (8). The former describes technologies that probe the captured object through physical contact. The latter can be further divided in two categories: non-contact active, which contains methods that acquire shape information by emitting and detecting a form of radiation and non-contact passive, which acquire a shape by detecting an ambient radiation. The multiple implementations of these methods have been presented with various successes and caveats.

Since the goals of this work included building a mobile hand held scanner, no further research was conducted on the contact –based technologies. Such a technology would obviously be difficult to implement for mobile devices.

2.1.1 RGB methods

The RGB based methods can be categorized under the non-contact passive. The methods rely on the existence of visible light when determining the shape of an object. Several methods have been proposed for this category.

Vouzounaras et al. propose a method for deriving 3D geometry from a single RGB image by using vanishing point detection (3). Their method relies heavily on known geometrical structures such as orthogonal planes and requires the presence of floor and ceiling and/or walls in the captured images. They are able to reconstruct objects with simple geometry like block shaped buildings and building interiors. Their method is, however, unable to construct objects with spherical qualities.

Zheng et al. devised a method for constructing 3D models from multiple RGB images without the need for calibrating the cameras prior to the imaging (23). They combine techniques such as

Harris corner detection and line detection for obtaining the object geometry. Their method, however, requires the use of four cameras and a turntable to provide movement for the object.

Going through the research of the RGB based methods a few caveats quickly arise. These technologies always require an additional computational step for depth detection. They also often require additional physical aids, such as a turntable or more than one camera. Some of these methods also rely on detecting the taught object geometries, which leads to results often not matching the real object geometry.

2.1.2 Stereographic methods

While the stereographic methods can be categorized under the non-contact passive with the RGB based methods, there is a distinction to be made. Stereographic cameras are usually defined to contain two closely positioned pre-calibrated RGB cameras. While the stereographic cameras are only now becoming available at consumer level mobile devices, a great deal of research has already been conducted on how to build a depth map from stereo view.

Huang et al. propose a method for 3D capturing using a stereographic camera and utilizing the shape-from-shading theorem (2). Like Zheng et al. (22), they use a turntable style solution for capturing all the perspectives. The core of their implementation relies on a feature point detection (i.e. detecting the corners of the object), which in return causes problems with objects with spherical surfaces. They also use image processing methods to extract the object from the background, but the details of this process are not disclosed. In practice, this method only works with primitive geometrical shapes lacking any curved lines or spherical surfaces.

Jiajun Zhu et al. use an array of the high-end digital SLR cameras to capture a HDR stereo pairs covering 360 degrees horizontal views (12). In their method, each of the eight cameras (four stereo pairs) capture 18 high resolution RGB images with different projected light patterns (produced with a flash) per capturing position. To capture a whole scene, their method sometimes requires several capture positions. Their method is able to produce high quality depth maps and textures of the scene with the trade-off of extremely high imaging and processing times going as high as 50 minutes per capturing position.

Moro et al. propose a stereographic method for acquiring depth data accompanied with a regions-of-interest algorithm for extracting trained statistical models of the object from the scene (11). The need for the object geometries taught beforehand limits the usefulness of this approach.

2.1.3 Time-of-flight methods

The Time-of-flight based methods probe the object by calculating the round-trip-time of a pulse of an emitted radiation (19), e.g. laser or infra-red light, and are therefore classified into the non-contact active method category.

Pheat et al. describe a method for using two laser diodes and a digital still camera. They use a turntable to provide a full view of the captured object. In a typical situation, their method needs 200-400 individual line scans to reconstruct a single object (19).

The laser based methods are the oldest and most well-known out of the existing depth scanning methods and provide the best accuracy. The downside, however, is the price overall, and the fact that inserting a sufficient amount of laser diodes for capturing a whole scene at once in a mobile phone is not a reasonable solution.

2.1.4 Structured light methods

The depth cameras using a structured light method usually contain a regular RGB camera together with an IR light emitter/sensor. The cameras with these qualities are often referred as RGB-D cameras, and are becoming available as consumer level products as well, one example being the Microsoft Kinect device. The RGB-D cameras can be categorized under the non-contact active.

Izadi et al. propose a method for using the Microsoft Kinect depth camera for capturing individual objects from larger scenes (16). In their method, the scene is first fully reconstructed after which the user moves the object-of-interest, which causes the implementation to separate the object from the rest of the scene. To archieve this level of interaction, their method is partially implemented using the Nvidia CUDA programming language to take advantage of the computing power of the high-end GPUs.

2.2 Point cloud registration

A full reconstruction of a physical object always requires multiple frames to be captured around it from different angles, the reason being that no matter how the camera is positioned, one or more faces of the object will always be occluded. This introduces a need for a method for tracking the transformation between the individual object captures (9). Several methods for tracking or calculating the transformations have been presented.

2.2.1 Sensor placement methods

A turntable based solution can be used by tracking the transformation between the individual frames by saving the rotation value of the table alongside with each captured frame (3, 19, 22). The downside of this approach is the need for an external apparatus for rotating the object.

The method of pre-calibrating the cameras can be used when multiple cameras are present. The cameras are placed in known positions and the transformation between them is solved before the actual capturing process (12). Since the cameras have to be positioned around the captured object and their position in relation to each other has to remain constant, this method is not well suited for mobile purposes.

2.2.2 Computational methods

The Iterative Closest Point (ICP) is an algorithm for calculating a 3D shape alignment (10). ICP can be used to calculate the camera transformation for each 3D frame by comparing the overlapping 3D points in the frames. ICP is used by Izadi et al., though they note that performing the ICP estimation on all the frames in real time is only feasible due to their GPU implementation (16).

3 USED TECHNOLOGIES

Multiple pre-existing tools, technologies and hardware were used during the implementation of this thesis. All software libraries, frameworks and applications used in the implementation are governed by an open source community. While the Microsoft Kinect sensor is a commercial product and its official Software Development Kit (SDK) is a proprietary code (i.e. closed source), an open source implementation of the SDK exists and is used in the software implementation of this thesis.

While some of the used libraries and technologies were choices made during the design of the implementation of this thesis (such as the Point Cloud Library), others were predefined requirements of the Chiru project.

3.1 Microsoft Kinect

Kinect is a RGB-D camera developed by PrimeSense, a company expertised in the high-performance 3D machine vision technologies, and licensed by Microsoft. Kinect is used on the XBOX-360 platform as a gaming controller capable of reading human bodies and gestures. Instead of the traditional time-of-flight methods (such as laser scanning), Kinect uses a structured light approach for creating depth maps. It uses an IR emitter to project a light pattern and an IR sensor to detect the deformations in the projected pattern for resolving the depth (17).

The result of combining the RGB and the depth feeds of the Kinect is a 640x480 resolution RGB-D image (i.e. point cloud) where each pixel, in addition to a color value, contains depth information when available. In an ideal condition, the depth feed has a resolution of 3mm. The RGB and depth cameras both work with the frequency of 30Hz (21). The Kinect sensor and its RGB and IR cameras and IR emitter can be seen in Figure 1.

FIGURE 1. Microsoft Kinect depth camera

3.2 Point Cloud Library

The Point Cloud Library (PCL) is a BSD licensed software library built for n-dimensional point clouds and 3D geometry processing (1). PCL is built to be fully compatible with the Robot Operating System (ROS) and has extensive use cases in robotics, machine vision and 3D geometry processing. PCL provides state-of-the-art algorithms for filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. In addition, PCL includes OpenMP (5) implementations of many of its algorithms for multi-core processing. PCL is supported and developed by an international community of robotics and perception researchers.

For the implementation of this thesis, PCL was chosen because of its extensive coverage over the algorithms required for the 3D object reconstruction, such as the algorithms for filtering, segmenting and registration. While there exists other libraries for these tasks, such as the MeshLab server (20), none of them provide as a vast coverage over all of the required processing methods.

3.3 RealXtend Tundra

RealXtend Tundra is a 3D freely available open source virtual world platform created and maintained by realXtend association. Tundra SDK is a framework that treats fundamental

elements of the virtual worlds as an add-in functionality, which makes it suitable for a wider range of virtual applications and spaces (6).

RealXtend project was started in 2007 by Juha Hulkko and has been in active development ever since. In 2011, realXtend association was founded to govern the development of realXtend project (24). As of 2012, seven entities from around the world are part of realXtend association (26). For the Chiru project, Tundra was chosen as a baseline for implementing new user interaction paradigms.

RealXtend Tundra consists of two major components: Tundra server and Tundra Viewer-Browser client software. In the core of the Tundra architecture there is the possibility to extend its functionality with independently functioning modules. The software implementation of this thesis is ultimately aimed to be integrated to Tundra as such a module. RealXtend Apache2-based license can be found in Appendix 3.

3.4 Qt

Qt is a cross-platform application development framework originally developed by a Norwegian company, Trolltech and later acquired by Nokia (22). The runtime binaries for the Qt framework are found for all the major desktop operating systems, such as Linux and Windows, and for certain mobile operating systems, such as MeeGo, Maemo and Symbian. RealXtend Tundra is built using the Qt framework, and therefore Qt and its extensive modules are made of use in the software implementation of this thesis.

3.5 Git

Git is a distributed, low-overhead version control system (VCS) used for collaborative development, originally developed by Linus Torvalds. Compared to other more traditional VCSs, Git has the advantage of speed and the fact that it is not dependend on a central server or network access. Instead, all developers have a complete copy of the source *repository* and its history. The changes made to these local repositories are then pushed to other repositories through the so called *commits* (18).

GitHub is a free hosting service for Git repositories. GitHub has web based user interface for managing projects and repositories. The code produced by the Chiru project is hosted by GitHub.

3.6 Code profiler

For conducting the performance tests on the software implementation of the thesis, a profiler was needed. Several tools designed for this purpose were evaluated. Most of the evaluated tools were able to provide a runtime per symbol and a complete call graph. In practice, they proved less than satisfactory for profiling the software implementation that uses the PCL, which in turn heavily relays on the boost software library.

After evaluating the traditional profiling tools, it was determined that a profiler capable of profiling the predefined blocks of the code was needed. However, no such tool was found. Therefore, the profiler was implemented as a side project of the thesis.

The implemented light weight profiler is capable of defining the profiled scopes using C-macros. After the execution of the profiled software terminates, the profiler outputs a call count, average time and total runtime per profiled block. The profiler is platform independent and can be used to profile any future projects requiring such data to be gathered. An example usage of the profiler can be found from Appendix 2.

4 IMPLEMENTATION

For achieving the goal of reconstructing a solid 360-degree point cloud presentation of a physical object, a processing chain, capable of separating the object-of-interest from the background data and merging multiple scans of the object together was implemented. The whole process of capturing the object-of-interest consists of several iterations of the processing chain (Figure 2).

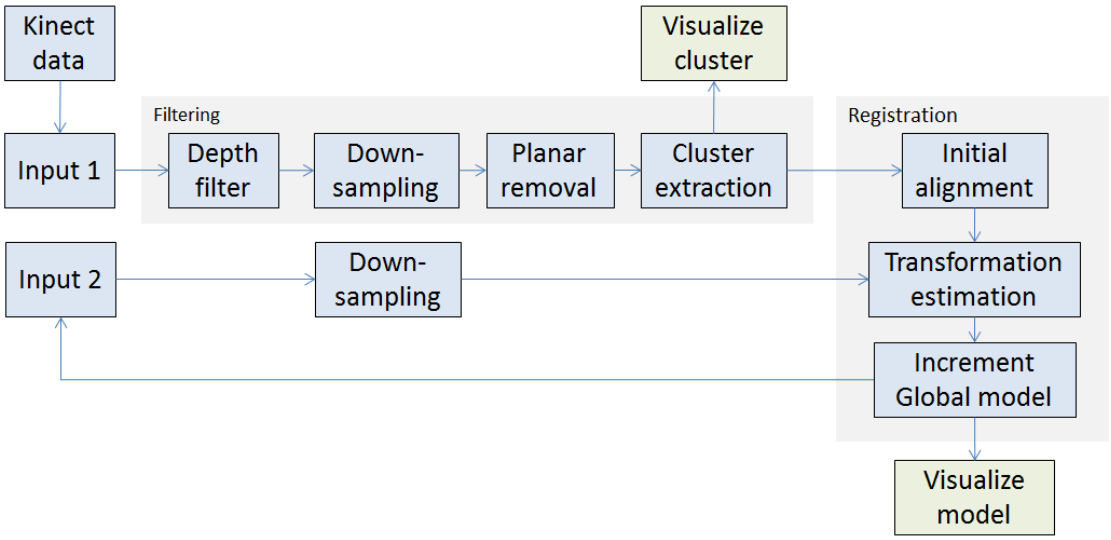


FIGURE 2. Overview of a single iteration of the data processing

The processing chain begins by obtaining a cloud of a input data from the Kinect sensor, which the user is holding and pointing towards the object-of-interest. The obtained input point cloud is then down-sampled, separated from the rest of the data and visualized for the user. The user is then given the option to merge the visualized object to the so called *global model*, which consists of all the previous merged captures. If the global model contains previously captured frames, a registration process is needed to estimate how much the Kinect sensor has moved between the current and the previous capture.

Once the user determines the global model sufficiently represents the actual physical object, the global model is transferred for the surface reconstruction process described in the bachelor thesis of Juha Hyvärinen (15). A full flowchart of the implementation can be found in Appendix 1.

4.1 Data acquisition

All of the data used for the object reconstruction is acquired from a Microsoft Kinect depth camera. Using the OpenNI toolkit the RGB and depth streams of the Kinect can be merged to build a point presentation (e.g. point cloud) consisting of 307200 individual points each with x, y, z –coordinates and RGB values. This point presentation can be visualized using the cloud viewer - utility of the PCL (Figure 3).

FIGURE 3. Unprocessed point cloud data acquired from the Kinect

While the Kinect sensor provides an accurate enough depth resolution for the object reconstruction, its depth sensor has problems with optical noise, shiny surfaces, occluded areas and flickering artifacts (27). Figure 4 depicts an error on the color acquisition on the object boundaries and visualizes the occluded area behind the toy horse.

The color error on the object boundaries is explained by the small shift between the depth and RGB images the Kinect sensor provides. This shift is individual to each Kinect device, and could only be overcome by implementing a separate calibration process (4).

FIGURE 4. Error in color acquisition

Currently, no methods for compensating the color acquisition error are implemented. This compensation should be implemented to improve the quality of the reconstructed objects.

4.2 Filtering

In the overall process of capturing and merging the multiple captured frames into a single object, data filtering is needed for two reasons:

- Segmentation: The captured object has to be separated from the rest of the data.
- Performance: The captured clouds have to be sub-sampled to achieve the real time performance.

Filtering is the limiting factor in the real-time visualization of the object-to-be-captured for the user. Once a new point cloud is obtained from the Kinect sensor, all the steps of the filtering chain are applied to the cloud. While the filtering process is ongoing, all new input clouds are discarded. After the filtering process terminates, the resulting cloud is visualized for the user and the system waits for the next input cloud from the Kinect. This limitation of the frame rate of visualizing the object only affects the process of aiming the camera towards the object-of-interest.

4.2.1 Depth filtering

While the Kinect sensor is capable of obtaining depth data on the range of 0.5m to 15m, errors in the depth estimation start to rise notably with the distance (4). Since the implementation of this thesis is specified for capturing small objects located on a planar surface, the noisy background data. can be discarded.

For filtering out the depth data, a simple passthrough filter is used. In the implementation of this thesis, the passthrough filter compares the Z-coordinate (depth) field of all the points in the cloud and drops all the points with the depth values above 1,5m. Figure 5 illustrates the original input data after the depth filtering.

FIGURE 5. Input data after the depth filtering process

4.2.2 Down-sampling

After depth filtering the input data, the data is down-sampled to ease the computational requirements of the latter steps. Uniform Sampling is used for this step of the filtering process. In Uniform Sampling, a 3D voxel grid (i.e. a set of 3D boxes) is created over the input data. After this, in each voxel all the points present are approximated with their centroid to produce a single point that presents the average of the voxel area (24).

4.2.3 Planar removal

Since the implementation of this thesis assumes that the captured objects are positioned on the floor, table or other planar surface, the points representing the object must be separated from the surface. Placing the physical object on a planar surface was a decision made during the implementation of this thesis to ease the process of object capturing. A version of RANSAC (Random Sample Consensus) fitted for 3D datasets is used for detecting the largest planar component and filtering it from the rest of the data. RANSAC is an algorithm for finding inliers in the sample data (13). In the implementation of the thesis, it is used for finding inliers for a planar model. Figure 6 illustrates the input data after removing the planar component.

FIGURE 6. Input data after planar removal

The implementation assumes that the floor or tabletop, on top of which the object-to-be captured lies, is the largest planar component present and leaves other planar surfaces intact. This leads to problems when trying to capture the object located near a wall or other large planar surface.

4.2.4 Clustering

After the planar removal, the input data might still contain unwanted objects. For filtering out the background objects, the remaining data is assumed to consist of object clusters. Each cluster is a collection of points located within 8cm radius of another point in the cluster.

For the actual implementation of the clustering, the Euclidean Cluster Extraction algorithm of the PCL library is used. The PCL implementation of the algorithm provides an extensive interface for segmenting and handling the clusters.

After the clusters in the input data are identified, the largest cluster is assumed to be the object-of-interest and all the other clusters are discarded. The remaining cluster is visualized for the user (Figure 7) who is then given the choice to integrate the cluster in to the global model.

FIGURE 7. The input data after the filtering process terminates

If the user decides to integrate the cluster to the global model, the registration process begins. If the user discards the data, the implementation waits for a new set of input data from the Kinect sensor and begins a new filtering process.

4.3 Registration

After the user chooses to integrate the clustered object to the global model, a registration process begins. During the first iteration of the registration process, the global model is still empty and therefore the captured cluster is moved to the global model as it is.

When the user decides to integrate another cluster, captured from a slightly different angle, to the global model, the real registration process begins. The captured cluster cannot be integrated to the previous as it is, because there exists a transformation that separates them. The registration process aims to resolve this transformation and rotate the captured cluster to fit the previously

captured clusters. The registration process is split into the steps of the initial alignment and the transformation estimation.

4.3.1 Initial alignment

The initial alignment step is only performed after the first two iterations of the registration process, the reason being that during the iteration I_0 there is no previous model, and during the iteration I_1 no previous transformation is estimated.

The aim of the initial alignment step is to move the captured cluster closer towards its actual alignment by applying the global transformation of the previous cluster to it. This helps the actual transformation estimation to reach the the convergence faster and more reliably.

The downside of this approach is that the user cannot freely capture the object from the the arbitrary angles and must instead capture successive frames by moving the camera around the object the in pre-determined direction.

4.3.2 Transformation estimation

After the transformation of the previous cluster is applied to the current cluster, the final transformation aligning the new cluster and the global model has to be estimated. For the transformation estimation, the Iterative Closest Point (ICP) algorithm is used.

ICP is an algorithm used to minimize the difference between two point clouds. The steps of the algorithm can be summarized as follows:

1. Associate the point pairs with nearest neighbor search.
2. Estimate the transformation using a least square cost function.
3. Transform the points using the estimated transformation.
4. Iterate until the termination criteria has been met.

The nearest neighbor search is used for finding point pairs during the ICP iteration. The search works by finding the nearest pair for a query point q in the d-dimensional metric space M from a set of points S when $q \in M$. In the implementation of the thesis, d is 4, consisting of the x, y and z-coordinates and the point curvature (point normal).

The ICP implementation of the PCL uses the Singular Value Decomposition (SVD) algorithm (7) for finding the least squares solution for the rotation matrix and translation vector between the corresponding point sets.

Once the transformation has been estimated, the target point cloud is transformed using the estimated transformation. At this point, the termination criteria are evaluated. If one or more criteria have been met, the point clouds are considered converged and the algorithm terminates. If no criteria are met, a new iteration begins by re-associating the point pairs. The ICP implementation of the PCL has several termination criteria, which are as follows:

1. The number of the iterations has met the preset maximum.
2. The difference between the previous transformation and the current estimated transformation is smaller than the preset value.
3. The sum of the Euclidean squared errors is smaller than the preset threshold.

In the implementation of this thesis, condition no. 1 is set to 120 iterations but is rarely reached due to condition no. 3, where the algorithm is terminated when the sum of the Euclidean squared errors falls below $1x10^{-4}$ meters. In the implementation of this thesis, condition no. 2 is ignored for the reason that during the implementation, conditions 1 and 3 were found to more accurately describe the converged solution.

Figure 8 depicts the alignment of two point clouds during the ICP iterations and finally reaching the convergence during iteration 12.

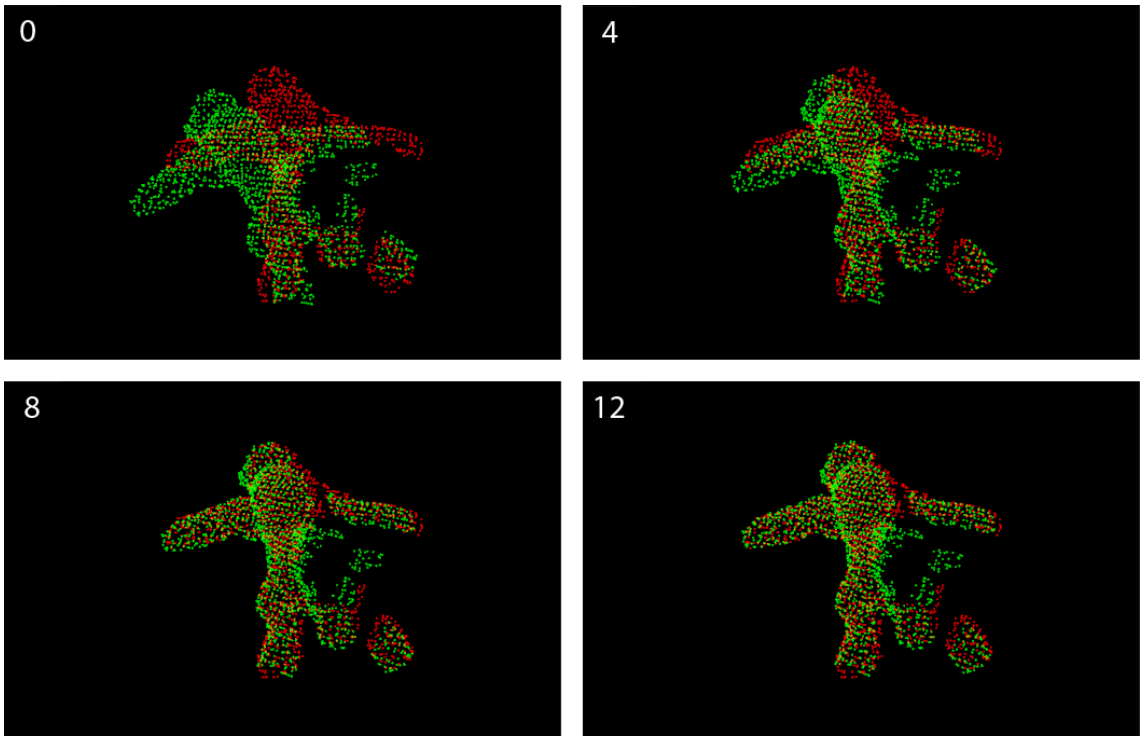


FIGURE 8. Alignment of two point clouds during ICP iterations

4.4 Incrementing the global model

Once the transformation aligning the current cluster and the global model is estimated, the cluster is merged to the global model. Since the cluster and the global model might still contain small errors in the alignment, simply adding all the points of the cluster to the global model causes some corresponding points to multiply in the global model.

To avoid this multiplication, an additional filtering step is applied after merging the two clouds. During this step the global model is down-sampled again using the Uniform Sampling method used for the initial down-sampling of the cluster. Using this method, all the duplicate points are averaged to form a single new point, and the global model more accurately presents the sum of the two point clouds.

After the new cluster is integrated to the global model and the filtering process terminates, the new global model is visualized for the user (Figure 9). After visually inspecting the model, the user can rewind it to a previous iteration, continue merging new clusters to it or finalize the model by starting the surface reconstruction process.

FIGURE 9. Global model constructed from 4 individual clusters

5 REALXTEND INTEGRATION

RealXtend Tundra has a support for extending its functionality with the so called *modules.* The software implementation of this thesis was integrated to Tundra as such module. The implemented module is called *ObjectCaptureModule* and in addition to the implementation of this thesis, contains the implementation described in another research (15). The overall architecture view of the integration can be seen in Figure 10.

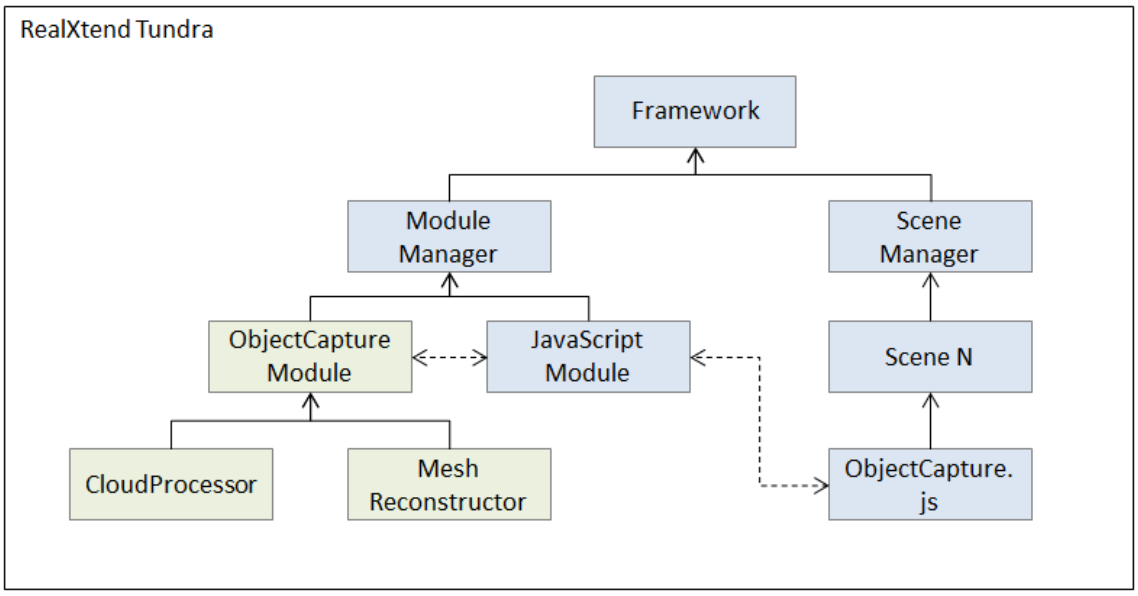


FIGURE 10. Simplified architecture view of the RealXtend integration

The implemented ObjectCaptureModule contains two sub modules: CloudProcessor, which in essence is the implementation of this thesis, and MeshReconstructor, which encapsulates the implementation described in the other research (15). The ObjectCaptureModule itself provides a public interface for the scripting purposes.

In realXtend Tundra, a scene is considered a collection of scripts and assets (i.e. 3D models, textures, sounds, etc.) which can describe a 3D virtual space or a 3D application. Usually scenes are located on a remote server and synchronized to the client through a network connection, but it is also possible to set up such a scene locally in the same device with the client. For the purpose of demonstrating the object capturing processing chain, a new local scene was created. The created scene contains a virtual screen, where the real time output of the Kinect sensor is rendered. Using this screen, the user can focus the Kinect sensor on the object-of-interest, and

perform the object capturing process. Once the user has decided to end the capturing process, the reconstructed point cloud, presenting the captured object, is transferred to the MeshReconstructor which creates a 3D mesh from it. This mesh is then visualized inside the Tundra scene, alongside to the virtual screen, at which point it behaves like any other 3D object in the scene (Figure 11).

FIGURE 11. A captured object and a virtual screen visualized inside Tundra scene

Since the ObjectCaptureModule itself defines no user interface, a Javascript application was implemented inside the Tundra scene. This application acts between the ObjectCaptureModule and the user, listening to the user input through the input API of Tundra and relaying the input to the ObjectCaptureModule script API.

6 RESULTS

The results of the thesis were evaluated from two perspectives:

- Performance: How well the implementation fulfills the real time requirements set for it.
- Visual correspondence: How accurately the reconstructed models correspond to their real life counterparts.

6.1 Performance

Because of the semi real time nature of the implemented method, the whole process of filtering and capturing the object was benchmarked for identifying the bottlenecks of the system. All the measurements were performed on a HP EliteBook 2760p laptop with the Intel Core i5-2540M processor and 4GB DDR3 RAM running Ubuntu 11.10 operating system.

6.1.1 Filtering

Figure 12 depicts the test settings for the filtering benchmarks. In setting A, a single object is processed. In setting B, the scene is cluttered with two smaller objects. In the last setting C, a larger object is added behind the object-of-interest to produce an incorrect object recognition.

FIGURE 12. Test settings for filtering performance benchmark

In settings A and B, the stuffed animal horse, being the largest cluster in the scene, is correctly identified as the object of interest and the rest of the data is discarded. In setting C, as is assumed, the group of paper trays is incorrectly chosen as the object-of-interest.

Figure 13 shows that while the depth filtering and planar removal times are largely independent on the amount of points in the cloud, the down-sampling and cluster extraction quickly start to suffer from the cluttered scenes. The total time for the filtering process varied from 101

milliseconds in setting A to 156 milliseconds in setting B, meaning that the frame rate for displaying the segmented object to the user varied from 6 to 10 frames per second.

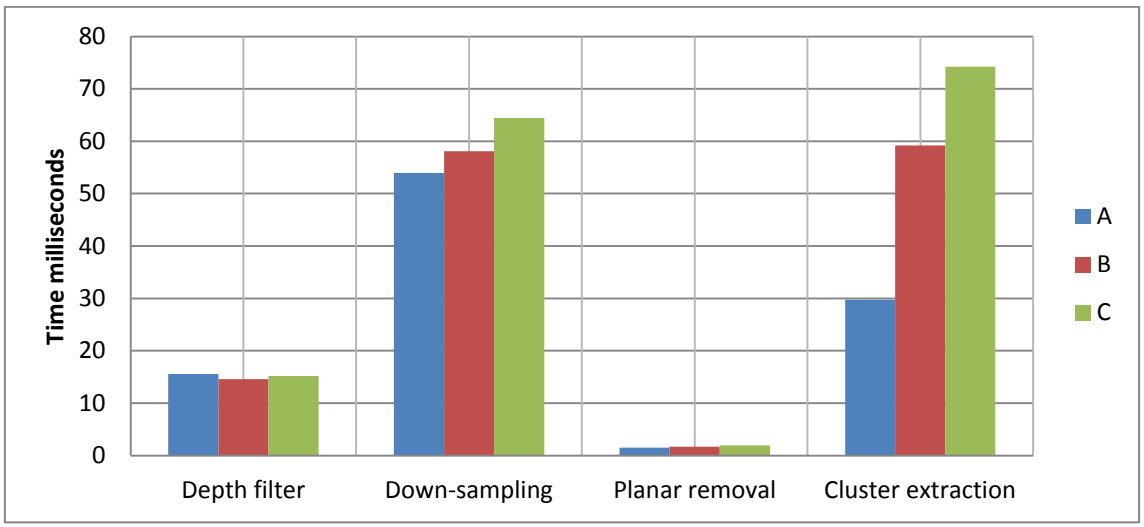


FIGURE 13. Average consumed time in different stages of the filtering process

Figure 14 depicts the average reduction of points over the stages of the filtering process. The input data is a 640x480 (307200) point cloud obtained from the Kinect.

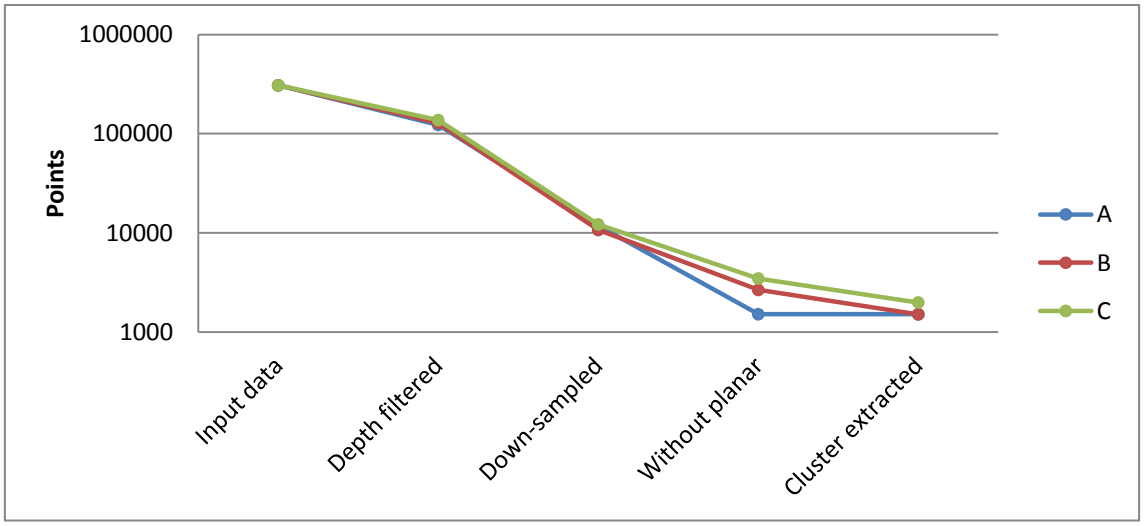


FIGURE 14. Average reduction of points over different stages of the filtering process

As shown in Figure 14, the point count after the depth filtering and down-sampling remains largely constant with small variations due to the noise in the Kinect depth data. In settings A and B where the stuffed animal horse was correctly extracted from the rest of the data, the point count

was reduced from the original 307200 points to approximately 1500 points, meaning a drop of approximately 99,5 percent.

6.1.2 Registration

For the registration benchmarks, two test settings were defined. In both settings, six point clouds of a single object were captured holding the Kinect freely in hand. For the first setting, a stuffed animal horse was chosen as the object. An office chair was used in the second setting.

Figure 15 depicts the accumulation of points in the global model. Both models show linear growth due to relatively constant difference in the angles between the captured clouds. Since the global model is always down-sampled after integrating new clusters to it, the duplicate points are never present in it.

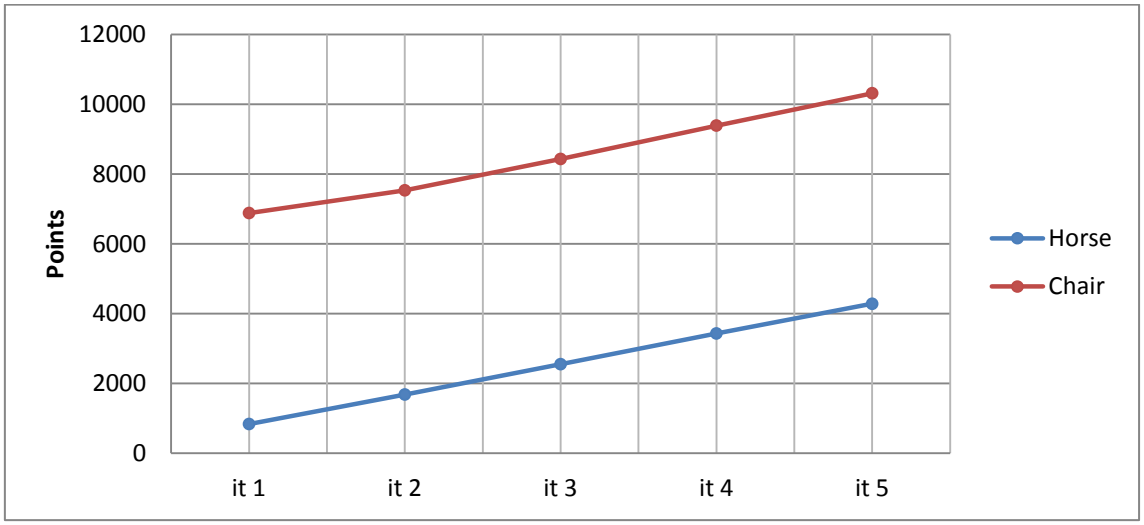


FIGURE 15. Accumulation of points in the global model

The time taken in estimating the point normal for the data sets (Figure 16) remains relatively constant during the iterations of the global model for both datasets, regardless of the increasing amount of total points as shown in Figure 15.

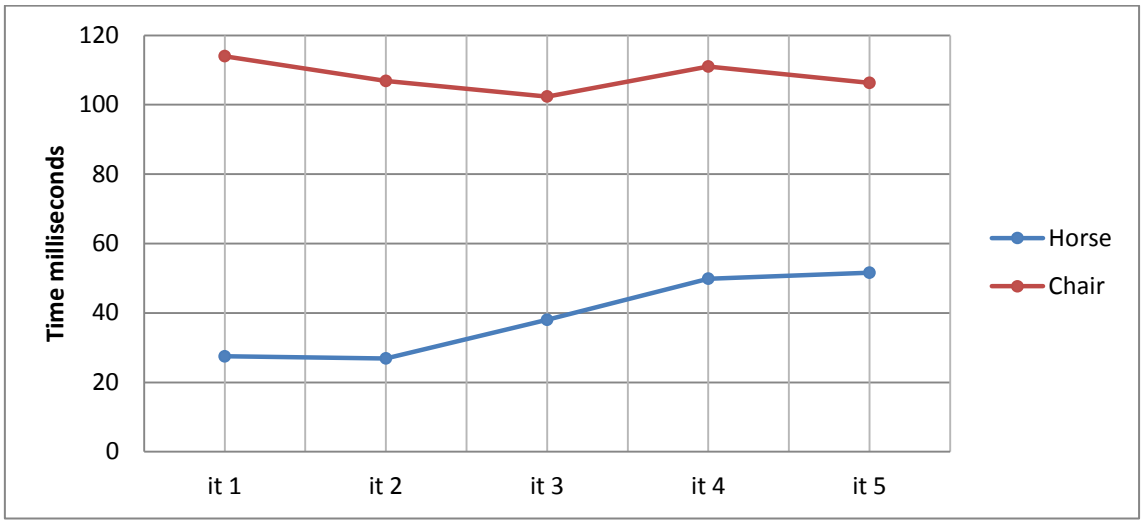


FIGURE 16. Duration of normal estimation

Figure 17 shows the running time of the ICP algorithm per global model iteration. Although, the number of ICP-iterations is capped, the computing time is relative to the number of points in the datasets. This increase of computing time per iteration could be overcome by registering each consecutive cloud only with the previous cloud (i.e. pair-wise registration) but this method has other drawbacks.

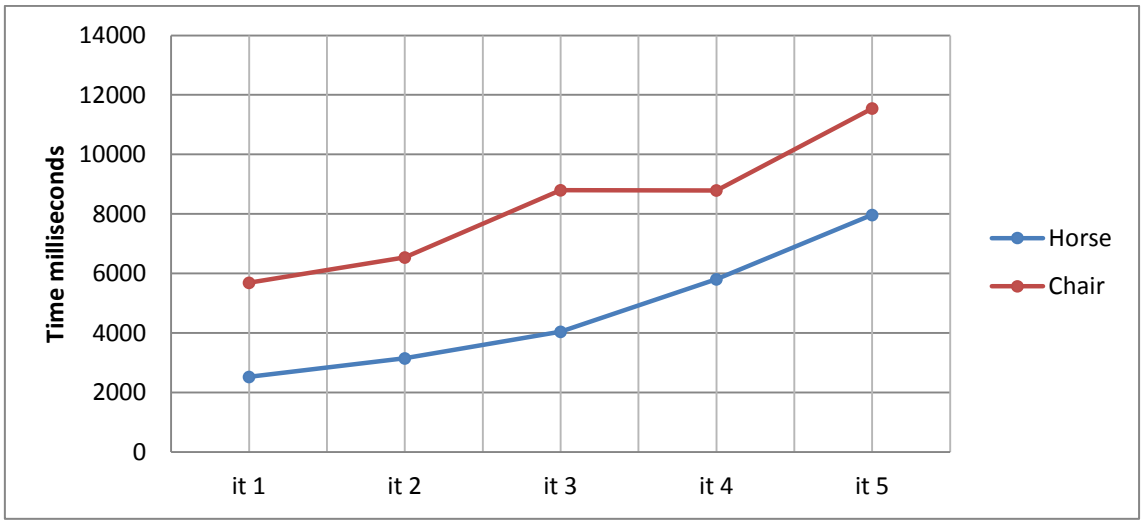


FIGURE 17. Duration of ICP transformation estimation

In average, a total of 4730 milliseconds were spent on the registration with the stuffed animal horse as the object-of-interest, the ICP transformation estimation spending the majority of the time with running time of 4690 milliseconds. In average, 8380 milliseconds were spent on

registering individual scans of the chair. The relation of the time taken for the normal estimation and the ICP transformation estimation in the case of the chair shows a similar relation to the case of the stuffed animal horse with the ICP transformation estimation spending the major part of the time with 8270 milliseconds.

6.2 Visual correspondence

For comparing the visual correspondence of the constructed point cloud presentations and the real world objects, two objects were chosen for inspection. First, a stuffed animal horse (Figure 18), for its complex geometry and a relatively small size and second, an office chair (Figure 19), for the large planar surfaces it consists of.

It should be noted that both figures only present the point cloud representations of the object, and not the final 3D meshes that are ready to be visualized inside the 3D graphics environment. The final 3D meshes, containing the solid surfaces, are presented in the bachelor thesis of Juha Hyvärinen.

FIGURE 18. Comparison of a stuffed animal horse and a constructed point cloud representation

As seen in Figure 18, the stuffed animal horse can be recognized from the point cloud representation constructed out of it. The constructed point cloud representation consists of four separate captures, aligned with each other. While the representation contains no significant color errors, small inconsistencies can be seen, especially in the head and the legs.

FIGURE 19. Comparison of an office chair and a constructed point cloud representation

The office chair, presented in Figure 19, contains more visible visual defects, compared to the stuffed animal horse, depicted in Figure 18. The actual real world office chair has shiny metallic legs. The Kinect sensor cannot acquire depth for reflecting surfaces, and no points without the depth data are integrated to the constructed point cloud model for obvious reasons. Another defect in the point cloud representation can be seen on the edges of the chair. The points on the edge of the point cloud presentation are colored white, unlike in its real world counterpart. This is due to the color acquisition error of the Kinect sensor on the object boundaries.

While the constructed point cloud presentation of the chair contains these visual defects, the actual real world object can still be recognized from it. As an interesting note, both large planar surfaces of the chair are left intact after the planar removal step of the filtering process, since the floor of the captured frames presented the largest planar surface, and was therefore removed instead.

Since the building of the model is done incrementally and not with the pair-wise method, the registration phase becomes more unreliable with each new cluster integrated to the global model. Figure 20 depicts the erroneously registered cluster in the global model.

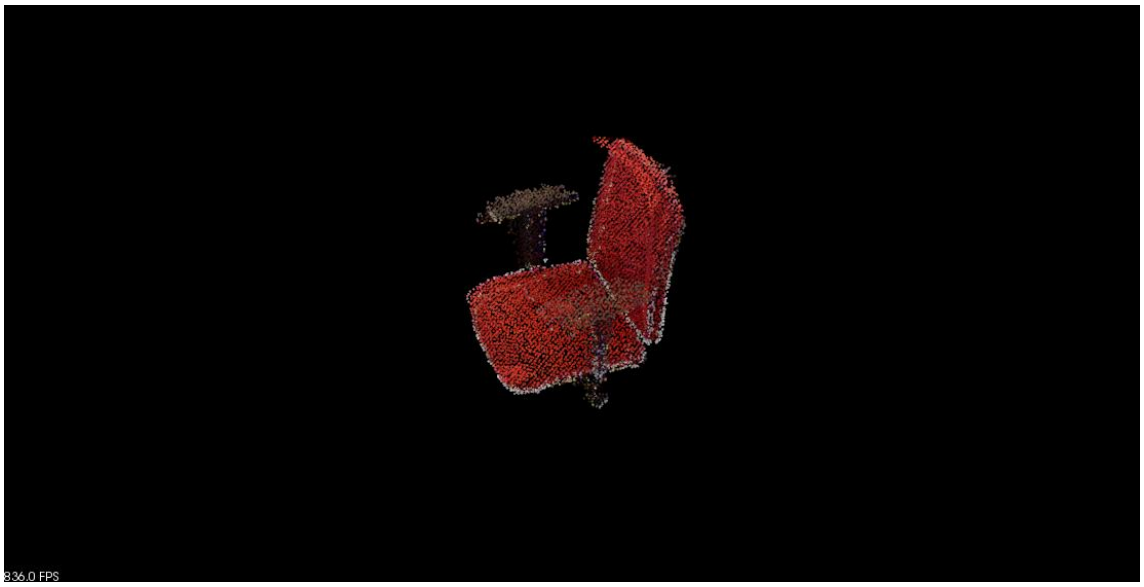


FIGURE 20. Erroneously registered cluster in the global model

As seen in Figure 20, the latest cluster, registered to the global model is not perfectly aligned with the rest of the model. The ability to rewind (i.e. remove the latest cluster from the global model) was implemented to overcome situations like this.

7 FUTURE WORK

While the method for down-sampling and segmenting an object by clustering the point cloud works in a sufficient frame rate, it has problems when the scene becomes cluttered and the object-of-interest is not actually the largest cluster present. In the future, this could be replaced by adaptive tracking methods where the system is taught the object geometry. Such methods are already implemented in the point cloud library (PCL.ORG).

In the registration, the devised method of overcoming the cumulative transformation error between the successive frames by building a global model causes certain problems. When the global model starts to grow, the geometry of the object starts to suffer. This in turn causes problems when point correspondences are tried to find between the new frames from the Kinect and the global model. This problem could be overcome by registering new frames only with the previous registered frame. However, this method would require a loop closure system to be built for compensating the cumulative error between the frames.

Most of the algorithms used in this work only utilize a single processor core. Although the majority of the current generation of mobile devices is not equipped with high-end graphical processing units like the desktop PCs, the multi-core CPUs are making their way in to these devices. Filtering and registration would benefit from parallelizing the processes and balancing the load for multiple CPU cores.

8 CONCLUSION AND DISCUSSION

The goal of the thesis was to implement a processing chain capable of constructing 360 degrees 3D point cloud presentations of the real world objects. This goal was achieved by implementing a processing chain capable of segmenting the object-of-interest from the rest of the data, and merging multiple captures, presenting the physical object, into a single point cloud presentation.

While the implementation presented in this thesis is capable of constructing presentations of the real world objects when the amount of captures stays relatively low, it still has problems when the amount of captures starts to rise. This problem could be overcome by switching the currently implemented incremental registration method to a pair-wise method, and implementing a bundle adjustment system to compensate for the cumulative alignment error, resulting from the pair-wise method.

Alongside with the method for constructing the point cloud presentations, an effort was made for integrating the system into the realXtend Tundra virtual world framework. It proves a decently working data path and a proof of the concept for the integration of a real-world object into virtual spaces, but for a complete end user implementation it would require further development.

Several problems and challenges presented themselves during the implementation of this thesis. Doing background research on the topic I had no previous knowledge on, was difficult at first. Devising a practical method for registering point clouds, based on the conducted research, proved to be challenging, especially when the real time and mobile requirements were established. Several methods for registering the point clouds and filtering the objects were implemented and evaluated, before successfully building a method that is capable of producing expected results. In the end, working on the topic of this thesis has personally taught me to better perform research on unfamiliar topics, and apply that knowledge in practice.

REFERENCES

1. 3D is here: Point Cloud Library (PCL). Robotics and Automation (ICRA), 2011 IEEE International Conference on; 2011.

2. 3D object model recovery from 2D images utilizing corner detection. System Science and Engineering (ICSSE), 2011 International Conference on; June; 2011.

3. 3D reconstruction of indoor and outdoor building scenes from a single image. Proceedings of the 2010 ACM workshop on Surreal media and virtual cloning New York, NY, USA: ACM; 2010.

4. 3D with Kinect. Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on; Nov.; 2011.

5. About the OpenMP ARB and the OpenMP.org. [Online]. Available at: http://openmp.org/wp/about-openmp/. Accessed Apr 2012.

6. Alatalo T. An Entity-Component Model for Extensible Virtual Worlds. Internet Computing, IEEE 2011 Sept.-Oct.;15(5):30.

7. Arun KS, Huang TS, Blostein SD. Pattern Analysis and Machine Intelligence, IEEE Transactions on title Least-Squares Fitting of Two 3-D Point Sets 1987 sept.;PAMI-9(5):698.

8. Curless B. From range scans to 3D models. SIGGRAPH Comput.Graph. 1999 November;33(4):38-41.

9. Bernardini F, Rushmeier H. Volume 21 (2002), number 2 COMPUTER GRAPHICS forum The 3D Model Acquisition Pipeline.

10. Besl PJ, McKay HD. A method for registration of 3-D shapes. Pattern Analysis and Machine Intelligence, IEEE Transactions on 1992 Feb;14(2):239.

11. Building virtual worlds by 3d object mapping. Proceedings of the 2010 ACM workshop on Surreal media and virtual cloning New York, NY, USA: ACM; 2010.

12. Fast Omnidirectional 3D Scene Acquisition with an Array of Stereo Cameras. 3-D Digital Imaging and Modeling, 2007. 3DIM '07. Sixth International Conference on; Aug.; 2007.

13. Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 1981 Jun;24(6):381-395.

14. Hulkko J. realXtend: Open Source Platform for Interconnected Virtual Worlds. 2011; Available at: http://realxtend.org/page.php?pg=contact. Accessed Apr 2012, 2011.

15. Juha Hyvärinen. Surface Reconstruction of Point Clouds Captured with Microsoft Kinect. Upublished.

16. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. Proceedings of the 24th annual ACM symposium on User interface software and technology New York, NY, USA: ACM; 2011.

17. Kinect: The company behind the tech explains how it works. [Online]. Available at: http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-the-company-behind-the-tech/. Accessed Apr 2012.

18. Loeliger, J. 2009. Version control with Git. O'Reilly Media: New York

19. Pheatt C, Ballester J, Wilhelmi D. Low-cost three-dimensional scanning using range imaging. J.Comput.Small Coll. 2005 April;20(4):13-19.

20. MeshLab. [Online]. Available at: http://meshlab.sourceforge.net/. Accessed Apr2012.

21. M. Tölgyessy, “The Kinect Sensor in Robotics Education,” innoc.at,no. Nov, pp. 1–4, 2010.

22. Nokia to buy Trolltech, will become a patron of KDE. [Online]. Available at: http://arstechnica.com/open-source/news/2008/01/nokia-buys-trolltech-will-become-a-patron-of-kde.ars. . Accessed Apr 2012.

23. A novel method for 3D reconstruction on uncalibrated images. Proceedings of the Third International Conference on Internet Multimedia Computing and Service New York, NY, USA: ACM; 2011.

24. PCL API Documentation. [Online]. Available at: http://docs.pointclouds.org/1.5.1/. Accessed Apr 2012.

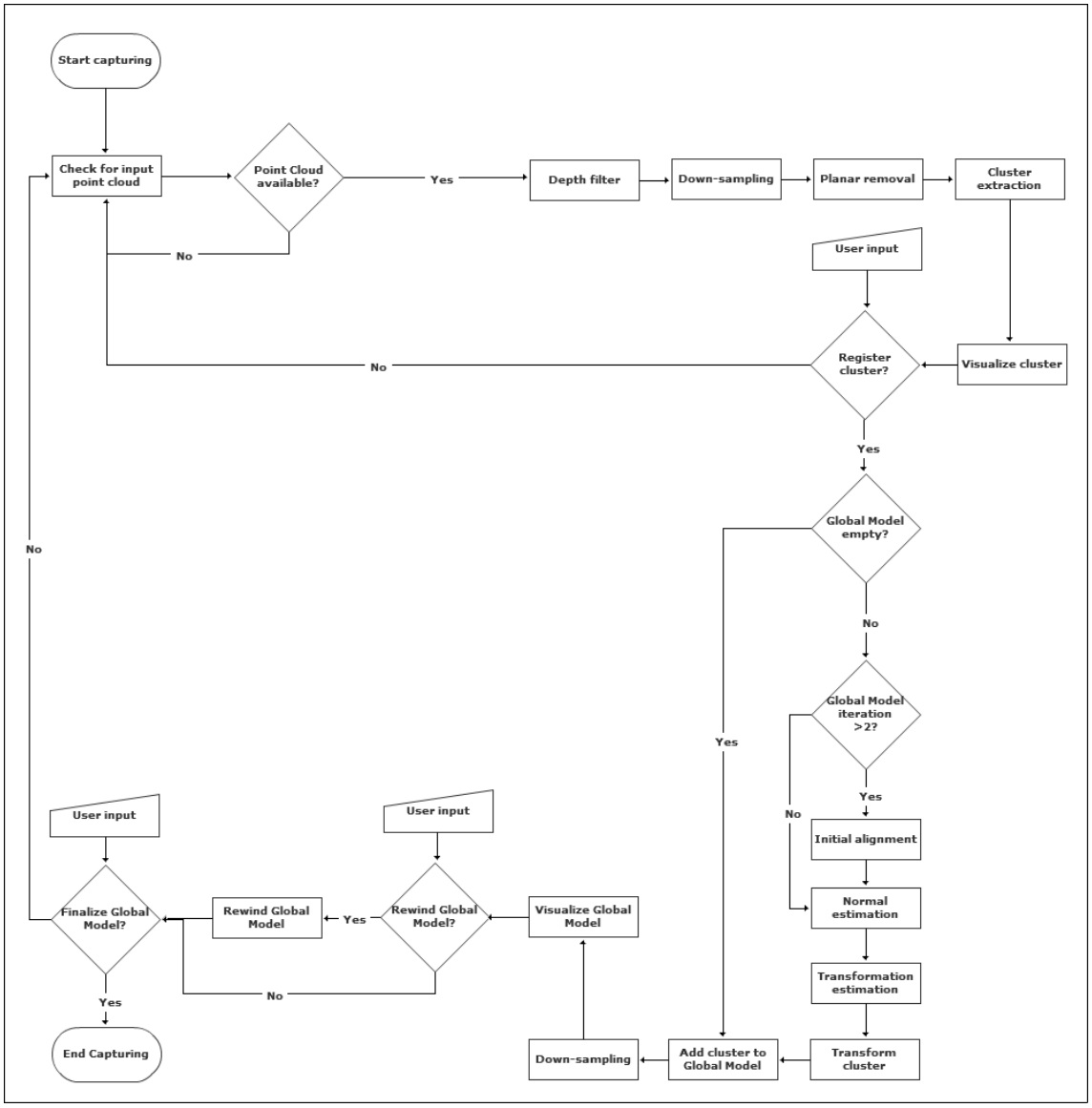
25. RealXtend Association formed. [Online]. Available at: http://www.hypergridbusiness.com/2011/04/realxtend-association-formed. Accessed Apr 2012

26. RealXtend association. [Online]. Available: http://realxtend.wordpress.com/realxtend-association/. Accessed Apr 2012.

27. Temporal filtering for depth maps generated by Kinect depth camera. 3DTV Conference: TheTrue Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011; May; 2011.

APPENDIX 1

IMPLEMENTATION FLOWCHART



APPENDIX 2

EXAMPLE USAGE OF THE IMPLEMENTED PROFILER

```
#include “cprofiler.h”

PROFILER_INITIALIZE(profilefilename.txt)

void task1()
{
   PROFILER_START_SCOPE(task1)
   for(int i = 0; i < 100; ++i)
   {
      sleep(1);
   }
   PROFILER_END_SCOPE(task1)
}

int main(int arch, char *argv[])
{
   PROFILER_START_SCOPE(main)
   task1();
   PROFILER_END_SCOPE(main)
   PROFILER_SHUTDOWN()
}
```

REALXTEND LICENSE

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other

modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or

contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the

above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.
END OF TERMS AND CONDIT