KINECT-BASED OBJECT RECONSTRUCTION

by

ANDREW R. PRICE

KENNETH G. RICKS, COMMITTEE CHAIR
JEFF JACKSON
MONICA D. ANDERSON

A THESIS

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in the Department of Electrical and
Computer Engineering in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2012

ABSTRACT

The Microsoft Kinect has recently grown to prominence as a widely used 3D sensor for both academics and hobbyists alike. This thesis presents a Kinect-oriented framework for reproducing physical objects using open or closed source software, GPU-accelerated hardware, and a 3D printer. Specifics of data capture, surface reconstruction, and manufacture are discussed, along with examples of reproducing mechanical and biological models. Finally, future systematic improvements and additional application areas are discussed.

LIST OF ABBREVIATIONS AND SYMBOLS

Voxel (**Vo**lumetric Pi**xel**): discrete spatial element

RGB-D (**R**ed, **G**reen, **B**lue, **D**epth): color and depth dimensions captured by Kinect

FOV (**F**ield **o**f **V**iew): Angular and radial region visible to sensor

(T)SDF ((**T**runcated) **S**igned **D**istance **F**unction): Popular format for storing surface/volume data

Variable Naming Conventions:

Symbol	Datatype
a	Scalar
$\boldsymbol{a}$	Vector
A	Matrix
$\boldsymbol{A}$	Tensor
$\boldsymbol{a}_n$	Scalar
$A_{m,n}$	Scalar
$\boldsymbol{A}_{m,n}$	Vector

ACKNOWLEDGMENTS

I would first like to thank those directly involved with producing the models and images used throughout this thesis: Emily Jones for her involvement with the Mako shark model, Hisham Ali for his implementation of the *ReconstructMe* application, James Yerby, Taylor Hall, and DJ Outlaw for volunteering as capture models, and Dr. Sazonov and Dr. Sharpe and the University of Alabama Honors College for the use of the 3D printers. I especially need to thank Dr. Ricks for all of his support in helping to convert a fun project idea into an academically rigorous undertaking, especially under tight time constraints. Many thanks to my other committee members, Drs. Jackson and Anderson, for proposing improvements to the process, as well as highlighting aspects of the project I had not considered. Finally, I would like to thank my parents for their unwavering love and support, and my friends for putting up with all of the late nights in the office, the odd weekend hours, and every time I said, "I can't, I have to finish my thesis."

CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS iii

ACKNOWLEDGMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

1 INTRODUCTION 1

2 CONCEPTUAL PRIMER 3

2.1 Scanning Technologies 3

2.1.1 LIDAR Scanners 3

2.1.2 Stereo Vision 4

2.1.3 Structured Light Scanners 4

2.2 Point Cloud Organization 4

2.2.1 Arrays 5

2.2.2 Octrees 5

2.2.3 KD-Trees 6

2.3 Basic Point Cloud Operations 6

2.3.1 Filtering 6

2.3.2 Normal Estimation 7

2.3.3 Segmentation 7
2.4 Fundamental Kinect Computations 8
2.4.1 Conversion to 3D 8
2.4.2 Surface Normals 9
2.4.3 Point Registration 10
3 RELATED WORK 11
4 PROCEDURE 13
4.1 Data Capture 14
4.2 Mesh Recreation 17
4.3 Object Reproduction 19
5 EXAMPLES 20
6 FUTURE WORK 26
7 CONCLUSIONS 28
REFERENCES 30

LIST OF TABLES

Table 1. Plane-to-Plane Accuracy Characterization Results 24

Table 2. Corner Radii for Fitted Surfaces 25

LIST OF FIGURES

Figure 1. Kinect FOV Frustum 8

Figure 2. Normal Vector for Noisy Planar Surface Derived from Adjacent Vectors 9

Figure 3. Data Flow with Possible Implementations 13

Figure 4. Depth Array of Model Airplane from Kinect 14

Figure 5. Stages in the Replication Process: Original, Point Cloud, Mesh, and Recreated Models 15

Figure 6. Results of Poor Mesh Reconstruction. 18

Figure 7. Scan of Aircraft with Semi-Transparent Wing Material 20

Figure 8. Mako Fin Reconstruction 22

Figure 9. Accuracy Characterization Structure with Planar Fitting Applied 24

Figure 10. Filleted Corners with Best-Fit Surface 25

CHAPTER 1

INTRODUCTION

In the little more than a year since its commercial release, the Microsoft Kinect has enjoyed great success in the hands of researchers and hobbyists seeking to use the economical 3D camera for more than the gaming platform it was intended to be. A primary factor in this success has been its low price point, around 20 times cheaper than competing technologies. In addition to this, the broad user base has contributed expertise for a wide range of applications, from robotic mapping to augmented reality. Coupled with the rise of additive manufacturing, commonly known as 3D printing, there exists the exciting capability to print a scaled replica of any physical object on the same order of size as a person. While the primary uses of such technology have mainly been as an interesting novelty to reconstruct faces or statues, the wide range of users and easy accessibility shows that this technology has powerful applications to academic research.

This thesis will present a generalized procedure for producing a scaled replica of a physical object, discuss a specific implementation of this procedure, and highlight some examples of research made possible with these replicas. The implementation presented makes use of existing tools to create an economical, straightforward procedure suitable for a large audience of possible users.

Chapter 2 will begin by covering some fundamental concepts necessary to understanding some of the behind the scenes work that the Kinect must perform to create a digital map of the world. This will lead into Chapter 3, which will discuss the recent work in the image processing field that has

enabled the real-time capture component used in this project. Next, Chapter 4 will discuss the general procedure for performing an object reconstruction, and will elaborate on the implementation of the procedure that was used to generate the current results. These results will be discussed in Chapter 5, which will cover a variety of objects that were scanned and printed, along with some lessons learned during the generation of these examples, as well as highlight some accuracy characterizations performed. Following that, Chapter 6 will discuss future improvements that could improve the quality of the reproduction process, as well as new directions that are made possible by this work. Finally, Chapter 7 will summarize the contributions outlined in this thesis, and will comment on some of their potential implications.

CHAPTER 2

CONCEPTUAL PRIMER

Before embarking on a discussion of the capture and manipulation of 3D data, it is necessary to cover some basic concepts to provide a framework and a vocabulary for discussing the mechanics of point cloud operations. This will allow the discussion of the central procedure in the following chapters to proceed directly, and will allow readers familiar to these concepts to skip to the next chapter.

2.1 Scanning Technologies

Obviously, the task of using digital means to reproduce a model of a real world object depends heavily on the ability to accurately reproduce a scene through digital means. To this end, a number of technologies have been developed to allow the capture of 3D scenes by the computer. In addition to these methods, a number of other ranging technologies exist, such as ultrasonic and infrared rangefinders. However, the resolution of these methods is significantly lower than their more powerful brethren, so they receive only a passing mention.

2.1.1 LIDAR Scanners

A LIDAR scanning system uses a swept infrared laser to measure points on the circumference of its scanning arc using the time of flight for the reflected laser light. This one-dimensional beam can be actuated along a second axis to produce a full 3D model of a scene. Laser scanners like this generally produce the highest resolution scans and the widest field of view (FOV) of these three techniques, but

require more time for a full 3D scan. They generally cost upwards of $1000, putting them out of range for many hobby and academic projects.

2.1.2 Stereo Vision

A stereo camera is able to measure the distance to a particular pixel by measuring the parallax shift between the two images of the scene. This gives the sensor a FOV similar to that of a standard camera, making it ideally suited for forward-facing applications on vehicles, as the region of interest (ROI) requires less time to scan than a full sweep from a LIDAR system. Stereo cameras can create RGB-D images, unlike laser scanners, which only calculate depth.

2.1.3 Structured Light Scanners

The Kinect belongs to a relatively new class of distance sensor that uses a structured light array to sense its surroundings. Structured light systems share some similarities with both laser scanners and stereo cameras. The sensor uses an infrared laser to project an invisible pattern onto its subject, which is then measured by an offset image sensor. This allows it to quickly generate an image that is more resilient to changing light and environmental conditions than a stereo camera system. Additionally, since they were developed as a consumer electronics device, they cost an order of magnitude less than their older siblings.

2.2 Point Cloud Organization

Point clouds are sets of data containing collections of 3D vertices, generally derived from an observation of a real world scene. They can be broadly divided into two categories: organized and unorganized. An organized dataset allows either a direct indexing or tree-based search to locate a specific point and, more importantly, its neighbors. This is crucial, as most meaningful point cloud

operations depend on being able to quickly compute properties about a point based on its k-neighborhood, or the k closest points. As such, an unorganized cloud should be parsed into one of the following tree structures to provide faster computations. There are several ways that this organization can be achieved, depending on knowledge of how the data was captured. The following sections owe much to the excellent tutorials provided by the Point Cloud Library, presented in [1].

2.2.1 Arrays

Arrays are the simplest form of organization for point data, and also one of the easiest to use. Data from a single observation or scan generally arrives as an array, with the array index representing some regular angular interval. The array can be single- or multi-dimensional, and successive sweeps by a laser scanner (each an individual one-dimensional array) can be combined into a 2D array if the angular step is known. It is important to note that a 1D or 2D array can still contain 3D points, the array simply describes the organization of those 3D points.

2.2.2 Octrees

An octree (octal tree) is a spatial hierarchy composed by taking a root volume element and splitting it in half parallel to all three axes. This produces a tree with each parent node containing eight equally sized children, with their position in space determined by the parent's location. Each leaf node contains a collection of all points that fall within its boundaries. The spatial resolution of an octree varies with the node depth, so regions with a higher point density can be subdivided further than regions with a low density, enabling fast k-neighborhood searches. In a sense, an octree is similar to a raster graphical format.

2.2.3 KD-Trees

If an octree is a raster representation of space, a KD-tree (k-dimensional tree, where k=3 for purposes herein) might be a vector representation. Instead of dividing the contained space into even portions, the KD-tree splits the points evenly between the two child nodes. This results in a roughly balanced binary tree, where each node contains the middle point that defines the dividing line. The line is drawn parallel to each of the three axes in rotation, so the first division might be along the X dimension, the next level might be along Y, and so on. In many circumstances, a KD-tree may provide faster search times than an octree, but is more difficult to update.

2.3 Basic Point Cloud Operations

Once a cloud has been organized, several characterizations derive easily from the organization and provide additional useful properties of the points and their neighborhoods. Like all basic operations, there exist more complex ways of performing the following procedures that are well suited for specific tasks. However, this section aims to simply provide a high level description of each process and the benefits each offers.

2.3.1 Filtering

Measurement error is inherent in any conversion from the physical world to the digital one. Range scanning is especially susceptible to this measurement noise, as readings may be altered by a surface's material or shape properties to give an inaccurate result. Thus, filtering noise is a fundamental and necessary operation on scanning data. Numerous point cloud filters exist, but the most straightforward approach is a statistical mean distance filter. In this operation, the mean distance to the other points in the k-neighborhood is computed for every point, and points with a mean distance greater

than some threshold value are discarded. This threshold can be set as a constant beforehand, or it can be driven by some statistical property such as the standard deviation of the mean distances.

2.3.2 Normal Estimation

Estimating the surface normal at a point is important for determining whether nearby points are part of the same object, or what underlying surface was being scanned. The formula for computing the normal vector to the plane defined by any three points is:

$$\boldsymbol{n} = (\boldsymbol{p}_2 - \boldsymbol{p}_1) \times (\boldsymbol{p}_3 - \boldsymbol{p}_1)$$

Given a particular point and its neighborhood, there are a few methods for estimating the normal vector. The simplest method is to average the normals generated from the cross products of the vectors to all of the other points in the neighborhood, but this is often unnecessarily computationally intensive to provide minimal additional accuracy. More often, a random subset of points is chosen for the cross product operation, sometimes with an additional cross product to verify consensus. Finally, if the points are indexed as a 2D matrix, it may be satisfactory to use only the neighboring points for a rough approximation.

2.3.3 Segmentation

Segmentation is the process that separates collections of points into the various objects they represent, such as floors, walls, or other objects. Segmentation is used extensively to help understand the content of a captured scene, rather than simply reproducing it in the computer. One common approach to segmentation is computing the derivative of the normal field to determine when one object stops and another begins. Alternately, pattern matching can be used to identify similarities to a predefined template such as a coffee mug or computer monitor. This thesis does not take advantage of segmentation

techniques, though perhaps a future implementation would use it to isolate the object of interest from the background.

2.4 Fundamental Kinect Computations

The basic point cloud operations outlined above have some specialized forms that allow them to be optimized for the real-time constraints of this operation. These optimizations will be outlined in Chapter 3, with mention of the papers that first outlined them. For now, the simple forms will be discussed to provide a frame of reference. For further information, please refer to [2].

2.4.1 Conversion to 3D

Data from the Kinect arrives as a "flat" 2D array of scalar depths. As mentioned in 2.2.1, the indices of the array correspond to a predefined angular interval that is determined by the intrinsic FOV of the

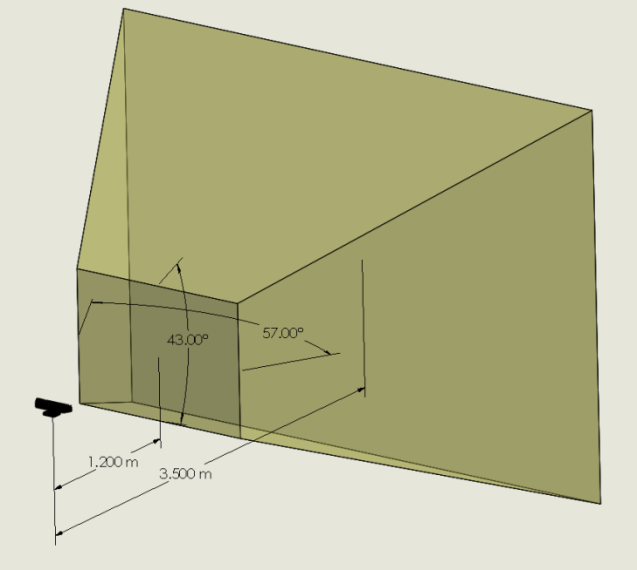

Figure 1. Kinect FOV Frustum

Kinect. In order to convert from the depth array to a true point cloud, a relatively simple function is employed. Let $\boldsymbol{D(t)}$ represent the depth image at time t. Now let $\boldsymbol{K}$ represent the intrinsic FOV tensor (matrix of vectors) of the Kinect. Each point $\boldsymbol{P}_{m,n}(t)$ can be computed as follows:

$$\boldsymbol{P}_{m,n}(t) = D_{m,n}(t)\boldsymbol{K}_{m,n}$$

where m and n are the row and column indices of the current point in the depth array and

$$\mathbf{K}_{m,n} = \frac{1}{sin(\boldsymbol{\phi}_m)cos(\boldsymbol{\theta}_n)} \begin{bmatrix} sin(\boldsymbol{\phi}_m)cos(\boldsymbol{\theta}_n) \\ sin(\boldsymbol{\phi}_m)sin(\boldsymbol{\theta}_n) \\ cos(\boldsymbol{\phi}_m) \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{sin(\boldsymbol{\theta}_n)}{cos(\boldsymbol{\theta}_n)} \\ \frac{cos(\boldsymbol{\phi}_m)}{sin(\boldsymbol{\phi}_m)cos(\boldsymbol{\theta}_n)} \end{bmatrix}$$

provided that

$$\begin{aligned} \boldsymbol{\phi}_m &= \frac{pi}{2} - c_\phi * (\frac{1}{2} - \frac{m}{r_\phi}), m = \{0, ..., r_\phi\} \\ \boldsymbol{\theta}_n &= c_\theta * (\frac{1}{2} - \frac{n}{r_\theta}), n = \{0, ..., r_\theta\} \end{aligned}$$

where *c* is the camera's viewing angle and *r* is the depth camera resolution, visible in Figure 1.

2.4.2 Surface Normals

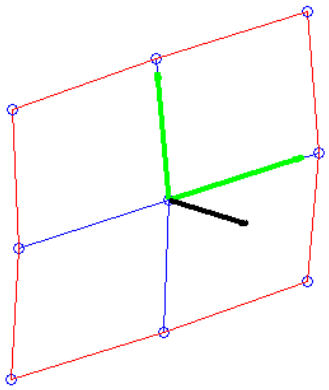

Figure 2. Normal Vector for Noisy Planar Surface Derived from Adjacent Vectors

Computing the surface normals is also a straightforward task, if one assumes that using the adjacent points for the cross product gives a sufficient approximation. With this assumption, the normal vector at every point reduces to

$$\boldsymbol{N}_{m,n} \approx (\boldsymbol{P}_{m+1,n} - \boldsymbol{P}_{m,n}) \times (\boldsymbol{P}_{m,n+1} - \boldsymbol{P}_{m,n})$$

2.4.3 Point Registration

The most computationally intensive phase in 3D mapping with the Kinect involves aligning successive individual scans into a unified global map. While several options exist for aligning related point clouds, the version employed here is known as iterative closest point (ICP). Traditional ICP is a cyclical process that contains four main steps. First, the incoming and existing datasets are processed to extract keypoints from the larger cloud. Next, the closest keypoint in the existing cloud to each keypoint in the incoming dataset is located, resulting in a set of keypoint pairings. The sum of the distances between each of the keypoint pairs is computed as the alignment error. Next, the linear system is solved to find the transformation that minimizes the alignment error. This process is iterated until the system error converges to an acceptable level.

CHAPTER 3

RELATED WORK

The problem of simultaneous localization and mapping (SLAM) in an unstructured environment is a well-traveled one in academics, and will be discussed only briefly here. SLAM focuses on solving the problem of concurrently computing new additions to a virtual map and the viewer's position within that space. One of the earlier projects to apply the principles of SLAM to the Kinect sensor was an MIT project using quadrotor helicopters for indoor mapping [3] based on algorithms presented in [4]. As a parallel effort, a number of companies began offering methods to convert the sensor's depth field data back into physical objects. Initially, this consisted of printing a body/face from one vantage point, requiring only one frame of data [5]. To create a full model, [6] used multiple Kinects to capture the subject from three different fixed positions. This created a complete object model, but required the Kinect captures to be serialized in order to prevent interference from the depth sensors. The method presented here combines these approaches to economically generate a full physical model using only a singular sensor.

The breakthrough most heavily relied upon in this implementation was presented by Microsoft Research in their KinectFusion technology demonstration, presented in [2] and [7]. KinectFusion allows for real-time 3D mapping and visualization by exploiting advances in GPU technology and accessibility, particularly with NVIDIA's CUDA architecture. The GPU is an ideal fit for processing the raw data

received from the Kinect, as each point in the 2D depth array can be converted to a 3D coordinate in the local reference frame independently from any other points.

In addition, the surface normals can be satisfactorily approximated in parallel by taking the cross product of the vectors to the adjacent points. This local observation data can then be integrated into the global map using a parallelized implementation of the iterative closest point (ICP) algorithm. In addition, the KinectFusion program attempts to model the underlying physical volume using a variation of signed distance functions (SDF) [8] called the truncated signed distance function (TSDF). However, the implementation presented here uses a point cloud representation of the observed objects as an intermediate data format, although it still uses the TSDF representation for the registration phase behind the scenes.

There are also other related projects dealing with the physical reconstruction of digitized objects. The original Microsoft implementation [7] includes a paragraph stating that the TSDF data was exported to a 3D printer, but few implementation specifics are included. In [8], the SDF data was used to produce a smaller version of a statue scanned using LIDAR. More recently, a commercial program called ReconstructMe [9] has been released that provides similar capabilities to the procedure presented herein. ReconstructMe uses the Kinect to capture the object, automatically segments it from the surroundings, and exports directly to a .stl file for a 3D printer. Finally, the popular MakerBot printer has been used in the media in conjunction with laser scanning to recreate busts and other objects [10].

CHAPTER 4

PROCEDURE

The procedure for rebuilding an object can be broken into three broad phases: point cloud capture, mesh recreation, and object reconstruction. In the generalized case, each phase is guided by an initial user input, though this step could be automated when the process is optimized for a specific application. Each phase primarily utilizes existing algorithms, tools, and techniques as building blocks to

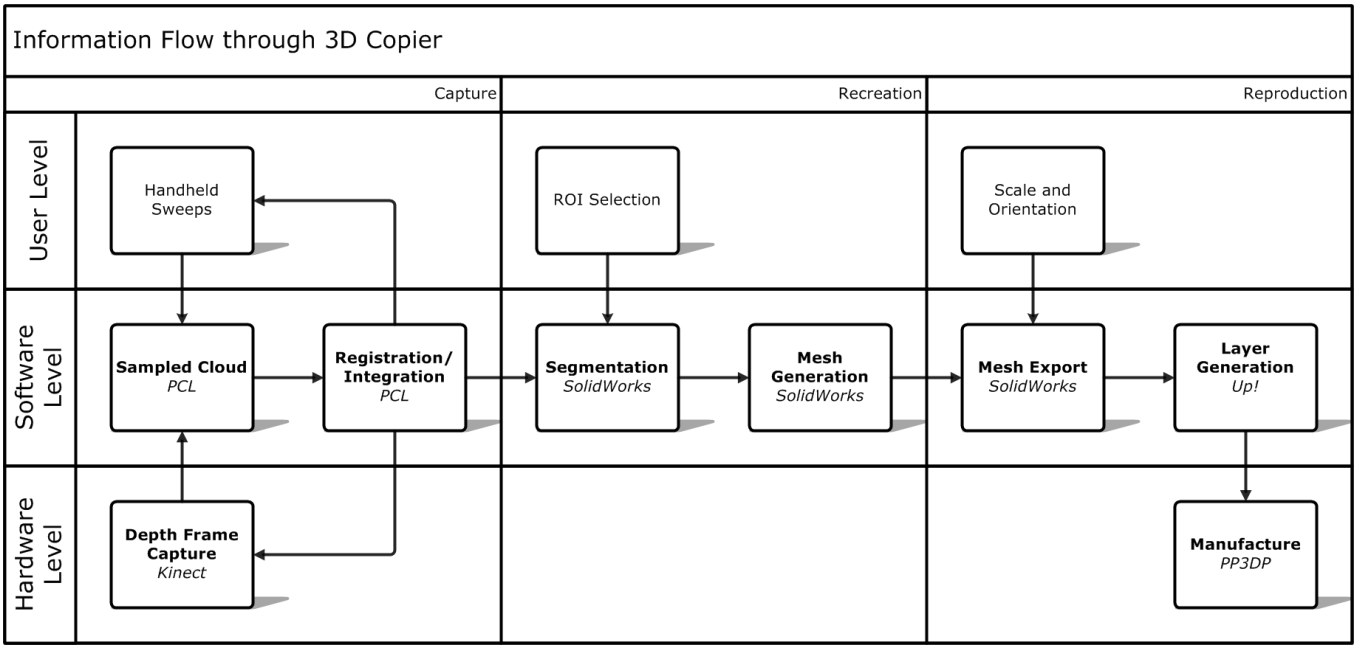

Figure 3. Data Flow with Possible Implementations

form an economical and straightforward workflow for creating 3D replicas. Where appropriate, special considerations for this specific implementation are discussed, mainly remedies for shortcomings due to hardware limitations. Referring to Figure 3, the steps of the process will now be discussed in greater detail.

4.1 Data Capture

The capture phase is an iterative process that is responsible for grabbing each depth frame

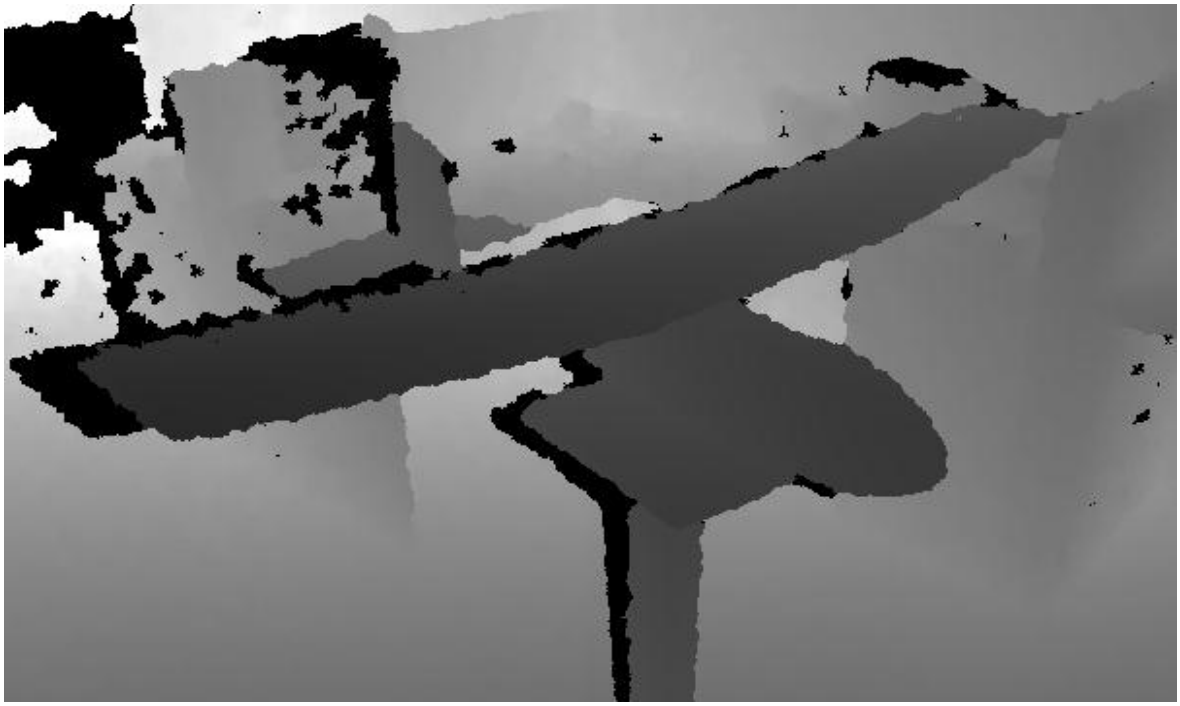

Figure 4. Depth Array of Model Airplane from Kinect

(Figure 4) from the Kinect and integrating it into the world already created by previous frames. In order to minimize development time, this project takes advantage of the open-source Point Cloud Library (PCL) [1] to handle the collection of data from the Kinect and the registration of each individual data frame against the existing model. PCL provides a variety of tools for capturing, manipulating, and visualizing point cloud data, whether from a Kinect or from a more traditional source like a LIDAR scanner. The real-time SLAM mapping module, called KinFu, is based on KinectFusion, and was written by Anatoly Baskeheev.

The KinFu implementation of KinectFusion provides an ideal framework for the capture phase. In addition to handling the raw data acquisition from the sensor and registering it against the global model, the KinFu application also generates several visualizations of the data being collected, including

the raw depth view, the shaded volumetric view, and the point cloud view. This information is shown to the user operating the camera, so that the camera can be reoriented to map any incomplete regions.

In order to maintain the real-time constraints on the system, the underlying TSDF volumetric representation is limited to what can be held in the GPU's RAM, as copying data to and from main memory proves to be fatally slow. This implementation uses a cubic volume 3000mm with 512 voxels per side, which results in around 1Gb of volume data, assuming double precision floating point is used (512^3 x 8 bytes). In theory, the physical size of the capture region can be reduced, allowing for greater resolution in the region of interest. However, experiments with this setup indicated that the Kinect's

Figure 5. Stages in the Replication Process: Original, Point Cloud, Mesh, and Recreated Models

scanning resolution is generally not sufficient to exploit these possible gains.

The movement of the camera during the scanning process plays a pivotal role in the success of a scan. The optimizations taken by KinFu assume a small incremental movement of the camera platform,

so the tracking will fail if the Kinect is not operated in a smooth manner. Similarly, while the tracking process does not explicitly extract keypoints like traditional SLAM, it does inherently rely on distinguishable features to be able to properly perform the ICP alignment. The practical consequence of this requirement is that scanning large planar or curved objects, or very slender objects, will often cause a tracking failure. These issues can be mitigated by adding other objects to the scene to provide a sense of reference.

The current capture procedure leaves particular opportunity for improvement in the hardware area. First and foremost, the limited resolution and significant minimum range of the standard Kinect proved to be formidable obstacles to scanning objects smaller than 10cm on a side. Unfortunately, this is the scale at which the replication process could truly excel. Most economy-level 3D printers have a maximum printable volume under $1ft^3$, so creating full size replicas is difficult when the scanner is targeted at objects at the 1-3m scale. However, Microsoft has recently released Kinect for Windows, which features a different set of optics to reduce the minimum range to be compatible with the typical distance of a user from their PC. Hopefully this will also translate to higher resolution at closer ranges, enabling new applications like the printing of mechanical replacement parts.

A second area for capture phase improvement is in the GPU component used for creating the real-time model of the environment. As mentioned, the size of the capture region (and thus, the complexity of the point cloud) is bounded by the GPU's RAM. For this experimental setup, a 2 Gb card with 196 parallel processing cores is employed. By comparison, the newly released NVIDIA Kepler K10 boasts 8Gb RAM with 3072 parallel processing cores [11], and these specifications will no doubt improve as upgraded graphics cards are released to market. Thus, future implementations should be capable of capturing much larger objects in greater detail.

4.2 Mesh Recreation

Once the capture phase has generated a satisfactory point map of the object's surface, the mesh recreation phase takes over the task of weaving the unordered points into a surface representation. The point cloud data is exported from the KinFu application to a .xyz file, essentially a tab delimited file with the 3D coordinates for each point stored in plain ASCII. While a binary format would be faster and more efficient, most commercial or open source tools for manipulating point cloud data accept the .xyz format, so its flexibility supersedes the performance gains of a binary intermediary format. The captured data can now be accessed by software such as AutoCAD, SolidWorks, or the freely available MeshLab. The implementation presented here uses the ScanTo3D addin for SolidWorks to handle point and mesh datasets (Figure 5).

After importing the dataset, the user may step in to focus the ROI. When scanning a particular object using the Kinect, the surrounding furniture, equipment, and bystanders will also be mapped, generating large quantities of extraneous data that are not directly part of the object in question. ScanTo3D includes a set of tools to automatically filter out sparse points, as well as manually crop the dataset. Once the dataset has been properly trimmed, the points are transformed into a mesh using an algorithm such as Marching Cubes [12] or Poisson reconstruction [13].

The ScanTo3D program is intended to generate a solid object model from the extracted mesh for use in further SolidWorks parts or assemblies. However, the program routinely failed to find fitted surfaces for the models scanned in these experiments, so converting from a mesh to a solid representation was not possible. Since most of the tools available in SolidWorks are strictly for solid modeling, this limits the ability of users to manually repair incompletely scanned models, although it does not hinder export to the printing phase, which uses a mesh-based format. Future implementations

should focus on flexible ways to heal incomplete models in order to avoid issues like those shown in Figure 6 where the 3D printed model of an office chair shows problems resulting from the scan and mesh recreation phases. Depending on the quality of the scan coverage, overly sparse sections of the cloud may be filtered out as noise, resulting in holes in the mesh surface. In the SolidWorks

Figure 6. Results of Poor Mesh Reconstruction.

implementation, the software attempted to heal these gaps, with varied success. Small holes were generally well handled, but a large hole on a concave surface often required a re-scan of the object. An opportunity for improvement would be the option to select the type of best-fit surface applied to cover the gap.

4.3 Object Reproduction

The final phase of the process is the reproduction phase, in which the new mesh structure is translated to a format for the 3D printer. The capabilities and costs of 3D printers and their associated software vary tremendously, so for this project two different printers were employed. The first printer was a hobby/prosumer model costing around $1,500 and capable of reproductions accurate to 0.2mm. The second printer was a much larger printer from Dimension with a larger printable volume (10x10x12in) and higher print resolution (0.1mm), but also with a much higher price point. Regardless of the hardware specifics, the de facto format standard for additive manufacturing is the stereo lithography (.stl) file, so the final stage can be implemented using nearly any hobby or professional level printer. Both of the printers employed on this project provided configuration software that could scale and position the model within the print area, as well as control the print resolution. The pricier model also provided options for setting the internal density of the printed model and for minimizing the amount of soluble support structure employed during the construction process.

For the models presented here, the construction process took 2-6 hours, depending on the size and complexity of the model, as well as the printer used. The final models ranged from 2-7 inches in their longest dimension, and were made of ABS plastic.

Future improvements in the object reproduction phase are tied to the printers. Additive manufacturing technology, while no doubt impressive, is still in its infancy. Printers with greater precision, larger print surfaces, and higher printing speeds will materially improve the replication process in the years to come.

CHAPTER 5

EXAMPLES

In order to demonstrate the range of applications of the procedure outlined above, three main types of models were processed: a model aircraft to demonstrate the ability to reproduce mechanical objects; a shark fin to represent biological objects; and several students to explore facial and body

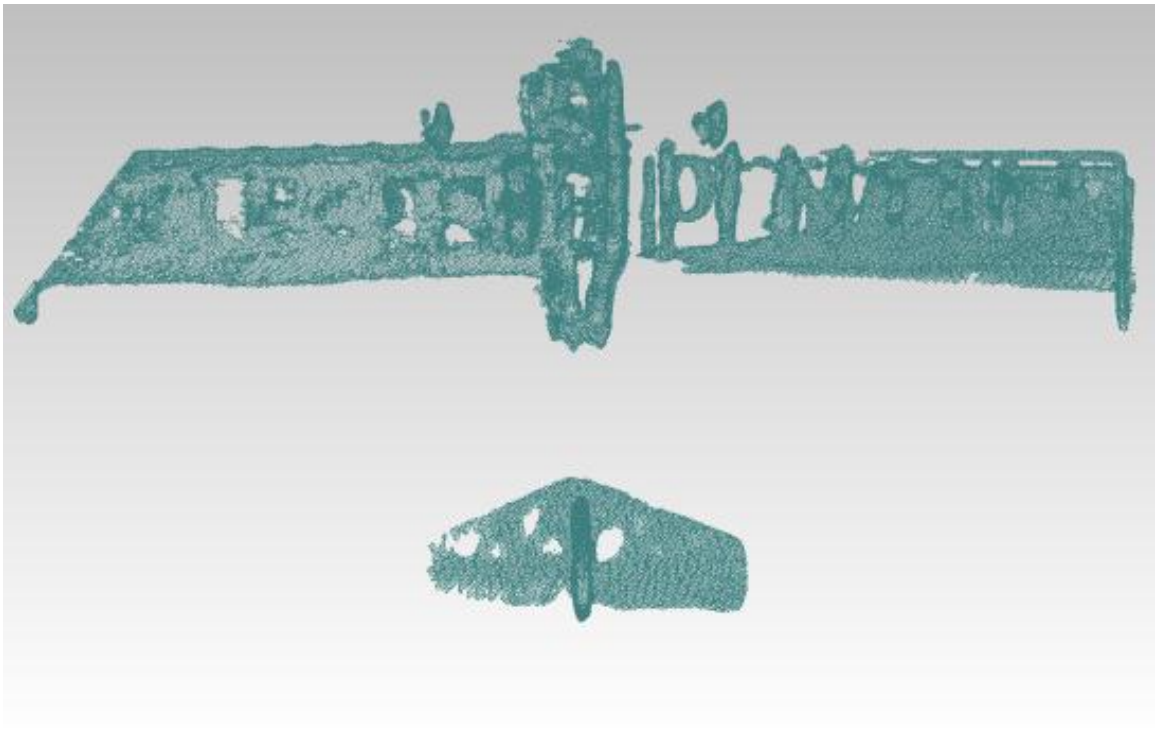

Figure 7. Scan of Aircraft with Semi-Transparent Wing Material

reconstruction. Each of these examples showed promising aspects, as well as some areas for improvement.

The first example processed was a medium sized RC aircraft used in a nearby research lab. With a wingspan of 51 inches, acquiring full coverage with the scanner proved to be the greatest difficulty, especially since the minimum depth requirement of the scanner strained the size constraints of the experimental setup (working from a cubicle using a USB extension cord). In addition, the Kinect has difficulty with curved, glossy surfaces, so scanning the tubular carbon fiber tail took several attempts to capture a good model. In Figure 5, one can also see the poor performance when scanning the plane's propeller. This occurs when the sensor observes the propeller from the front or back, but loses it when tracking from the side. The filter then interprets those points as noise due to their inconsistent visibility and eliminates them.

In addition to this first plane, replicating a similarly sized aircraft was also attempted. Unfortunately, the plastic wing material proved to be largely transparent to infrared, resulting in the poor model shown in Figure 7. These tests show that the scanning device responds poorly to reflective or transparent surfaces, and can also be influenced by ambient light sources in the room.

The next phase of testing focused on a mako shark's pectoral fin used by one of the aerospace

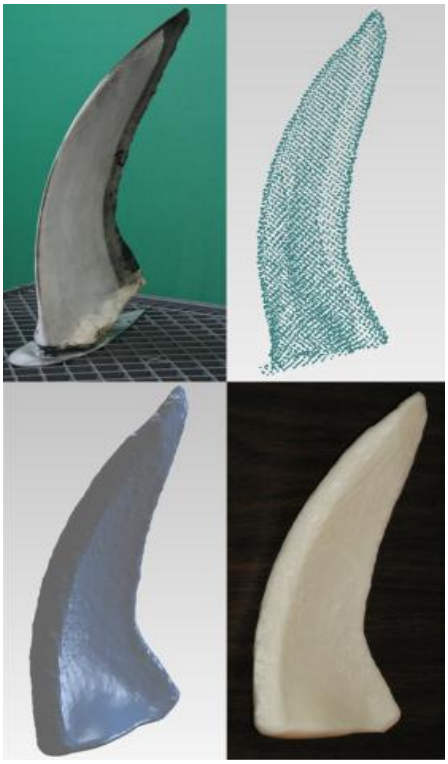

Figure 8. Mako Fin Reconstruction

labs. The fin, at around 1.5 feet in length, proved to be an ideal size and texture for the Kinect to capture. However, the fin tapers to a slender trailing edge, which the algorithm seemed to find difficult to capture. One of the goals for this capture was to be able to compare the drag of the original textured fin to that of a completely smooth replacement. However, the surface produced by the lower-cost printer proved too rough for a valid comparison. Sanding or coating the printed part with a smoother material would likely provide a more accurate reconstruction.

Finally, the procedure was demonstrated on student volunteers. Capturing human shapes proved nearly ideal, since that was the original purpose of the Kinect, after all. The body has few sharp corners and clothes rarely have the reflective or transparency difficulties imposed by many plastics or metals. However, loose fitting clothing did seem to pose more of a problem than more fitted designs, as the

loose material had a tendency to move somewhat independently of its wearer, forcing the filter to rebuild some surfaces and occasionally lose its bearings on the subject.

All of these successful examples passed the subjective observational test in that they looked very similar to their source models. However, for this procedure to truly advance into the research or manufacturing realms, a more empirical testing method is required. This thesis does not attempt to formulate a comprehensive end-to-end characterization for reproduction accuracy, although this would be an ideal front for continued development. Instead, the following methods were used to calculate an expected effective model capture accuracy, with the full system's fidelity being additionally determined by the printer used.

Of principal interest was what feature size the Kinect scanner was capable of accurately recording. To facilitate this, several test structures were scanned and imported into SolidWorks (Figure 9). Planar faces in the reconstructed mesh were approximated by a planar best-fit surface, in order to more acurately measure the distances involved. These measurements were then compared against the real-world values, and the absolute and percentage errors were computed. The average plane-to-plane error for the systems measured was 0.51cm. The full results are presented in Table 1.

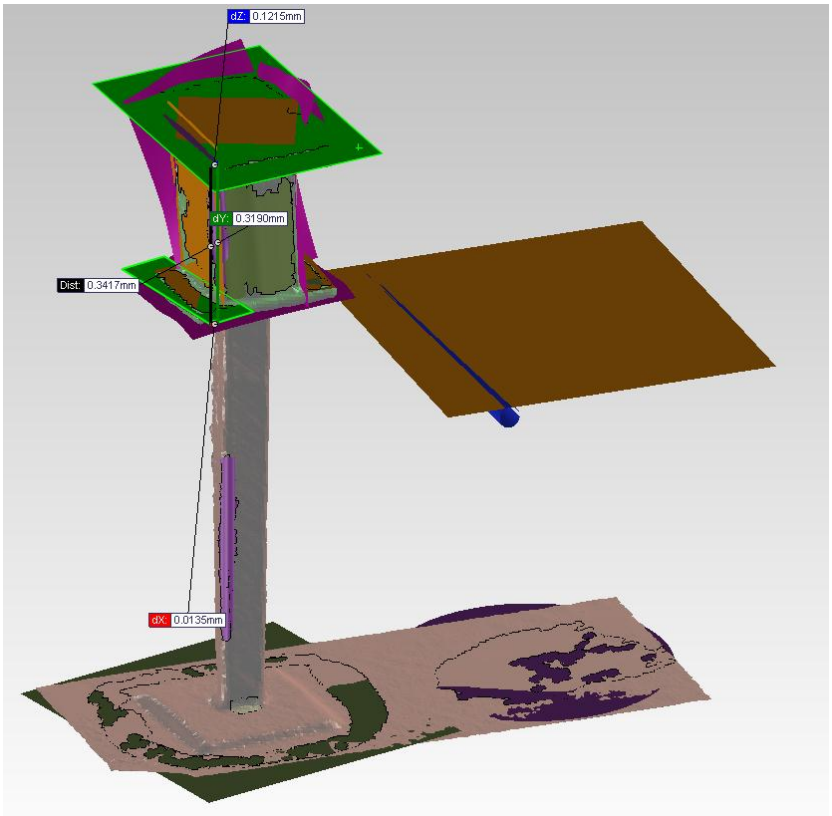

Figure 9. Accuracy Characterization Structure with Planar Fitting Applied

Table 1. Plane-to-Plane Accuracy Characterization Results

Measurement	Measured (cm)	Actual (cm)	Error (cm)	Percent Error
Test Object A	4.60	3.81	0.79	20.73%
Test Object B	6.06	5.08	0.98	19.29%
Test Object C	7.95	7.62	0.33	4.33%
Test Object D	14.73	14.92	-0.19	-1.29%
Test Object E	16.51	16.03	0.48	2.97%
Test Object F	34.58	34.61	-0.03	-0.08%
Test Object G	73.30	72.71	0.59	0.81%
Test Object H	88.96	89.69	-0.73	-0.82%

In addition to the measurement error, a second issue affecting the reproduction accuracy is that reconstructed meshes do not have perfectly sharp edges. Instead, the meshing process tends to automatically fillet corners, which limits effective feature size even more than the linear error. Applying a cylindrical or revolve best fit to the corner surfaces resulted in an average fillet radius of 2.13cm for the six different test objects shown in Table 2, far more significant than the plane-to-plane error. Certainly a higher resolution capture would aid in solving this issue, which currently determines the lower bound on the replication process's size capabilities.

Table 2. Corner Radii for Fitted Surfaces

Corner Radii (cm)
1.99
2.16
2.05
2.13
2.28
2.17

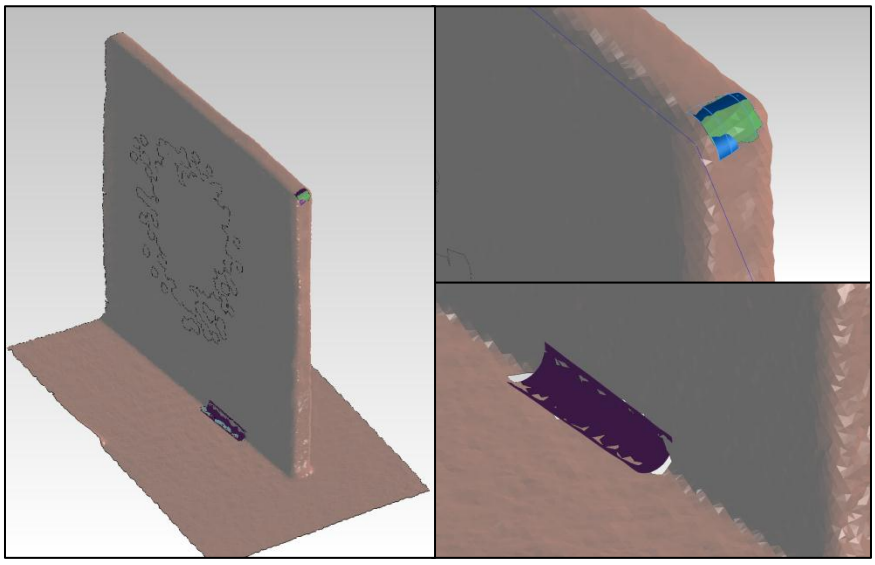

Figure 10. Filleted Corners with Best-Fit Surface

CHAPTER 6

FUTURE WORK

In addition to the advances in the process for creating 3D replicas outlined previously, there are a number of yet unexplored applications awaiting experimentation. One such area might be the field of paleontology. Large fossilized bones are at an ideal scale for capture with the Kinect scanner. The process for transporting discoveries from the dig site to a research facility is a labor intensive and risky procedure, generally involving coating the newly unearthed fossil in plaster of Paris. The procedure outlined would allow the new fossil to be digitized on site without worrying about damage from the shipping process. Once digitized, the new fossil could be replicated at a variety of locations, opening up new research avenues for distributed facilities and educational opportunities for schools, who could easily print a copy of a new discovery to show students.

Another exciting possibility lies in the world of manufacturing. The ability to create high-fidelity replacement parts on site would certainly revolutionize the repair and restoration industries, as well as raise difficult questions for the handling of intellectual property rights. And when we do travel back to the moon or on to Mars, it would not be difficult to imagine 3D printers creating new parts from in situ resources based on models captured on Earth and transmitted to the remote outpost.

The general scanning and capture procedure could also significantly advance some areas of medicine. For instance, the existing implementation could expedite the production of customized prosthetics. Replacing the Kinect input with a different input device such as an MRI could enable the

production of individualized joint replacements. A different 3D printer might allow the production of scaffolding on which to grow an exact copy of a failing organ. Finally, orthodontic work might no longer require time-consuming molds to build accurate models of the teeth and gums.

CHAPTER 7

CONCLUSIONS

In summary, a method has been presented to economically convert a physical object into a digital representation and then back to a real world object. While not a unique process in itself, the implementation presented uses existing tools and the inexpensive Kinect sensor to create a straightforward and economical process that is suitable for many users. Perhaps even more exciting, many of the procedure's constituent components are very active fronts of development, meaning that scanning resolution, real-time data acquisition and integration, and reproduction faithfulness are sure to improve dramatically over the coming years.

In addition to the exciting technical feats made possible by these tools, some consideration should be given to repercussions outside the strictly technical arena. The ability to quickly duplicate plastic components is sure to raise some challenging intellectual property issues, even as it opens up great new possibilities for the end user. Likewise, companies may take additional steps to protect trade secrets from prying eyes, as reproduction technology could accelerate a rival's ability to catch up to new developments.

In the end, however, the benefits of easy 3D reproduction will outweigh the reasons for concern. The ability to create exact replacement parts and easily repeatable test structures ought to be a great benefit to all branches of engineering. Educators could also benefit from printing classroom models taken from current subject matter. Artists and architects could easily incorporate copies of real-world

scenes into their creative work. Finally, using a different scanning technology such as MRI, physicians might be able to produce fully customized joint replacements and prosthetics. Really, the applications are only limited by the imagination (and the size of your GPU).

.

REFERENCES

[1] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[2] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *International Symposium on Mixed and Augmented Reality*, Basel, CH, 2011.

[3] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox and N. Roy, "Visual Odometry and Mapping for Autonomous Flight using an RGB-D Camera," in *International Symposium on Robotics Research*, Flagstaff, AZ, 2011.

[4] P. Henry, M. Krainin, E. Herbst, X. Ren and D. Fox, "RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments," in *International Symposium on Experimental Robotics*, New Delhi & Agra, India, 2010.

[5] "3D Printing with Kinect," MakerBot Industries, LLC , 26 May 2011. [Online]. Available: http://www.makerbot.com/blog/2011/05/26/3d-printing-with-kinect/. [Accessed 22 February 2012].

[6] blablabLAB, "Be Your Own Souvenir!," Vimeo, 29 March 2011. [Online]. Available: http://vimeo.com/21676294. [Accessed 22 February 2012].

[7] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison and A. Fitzgibbon, "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera," in *ACM Symposium on User Interface Software and Technology*, Santa Barbara, CA, 2011.

[8] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *SIGGRAPH*, New Orleans, LA, 1996.

[9] C. Heindl, "ReconstructMe," [Online]. Available: http://reconstructme.net.

[10] "Bre on the Colbert Report," MakerBot Industries, LLC, 9 June 2011. [Online]. Available: http://www.makerbot.com/blog/2011/06/09/bre-on-the-colbert-report/. [Accessed 3 March 2012].

[11] NVIDIA Corporation, "NVIDIA Tesla Kepler GPU Computing Accelerators," May 2012. [Online]. Available: http://www.nvidia.com/content/tesla/pdf/NV_DS_TeslaK_Family_May_2012_LR.pdf.

[Accessed 28 June 2012].

[12] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *SIGGRAPH*, Anaheim, CA, 1987.

[13] M. Kazhdan, M. Bolitho and H. Hoppe, "Poisson Surface Reconstruction," in *Eurographics Symposium on Geometry Processing*, 2006.