RWTH AACHEN UNIVERSITY

FabScan

Affordable 3D Laser Scanning of Physical Objects

Bachelor's Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Francis Engelmann

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr.-Ing. Kowalewski

Registration date: Sept 02th, 2011
Submission date: Sept 30th, 2011

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, July 2011
Francis Engelmann

Contents

List of Figures

List of Tables

Abstract

This thesis presents a hardware and software system for digitalizing the shape and color of physical objects under known environmental conditions.

The proposed system is conceived as a "Do-It-Yourself 3D Laser Scanner". Anyone with enough interest, can built his own FabScan 3D-Scanner. As such, the project tries to support Gershenfeld's vision of *Personal Fabrication* [Ger05].

The setup employs an ordinary webcam, an affordable line laser, two stepper motors and an Arduino Uno [Uno11]. The FabScan software is implemented using OpenCV [BK08], the Powercrust algorithm [AC01], C++ and Objective-C.

The software is capable of mapping a colored point-cloud from the object, transforming the point-cloud into a surface-mesh and converting the surface-mesh into STL, which is a stereolithography file format used among others for 3D printing [FF11]. Post processing with additional software, such as MeshLab [CR08], is no longer needed.

I have discussed the challenges faced during the construction of the different prototypes, the employed solutions and I am giving an evaluation of the final prototype.

Finally, the reader is provided with detailed blueprints allowing the replication of the complete system for less than 130€.

Überblick

In dieser Arbeit beschreibe ich die Hard- und Software eines Systems das der Digitalisierung von Form und Farbe physikalischer Objekten unter bekannter Umgebung dient.

Das vorgestellte System ist als ”Do-It-Youself” Projekt gedacht. Jeder mit genug Interesse, kann seine eigenen FabScan 3D-Scanner nachbauen. Somit unterstützt das Projekt auch Gershenfelds Vision des *Personal Fabrication* [Ger05].

Der Aufbau basiert auf einer gewöhnlichen Webcam, einem erschwinglichen Laser, zwei Schrittmotoren und einem Arduino Uno. Die FabScan Software ist geschrieben unter Benutzung von OpenCV[BK08], dem Powercrust Algorithmus [AC01], C++ und Objective-C.

Die Software kann farbige Punktwolken von Objekten erzeugen. Diese Punktwolken können in ein Oberflächen-Mesh umgewandelt werden und schliesslich als STL exportiert werden. STL ist ein stereolithisches Datei Format welches unter anderem beim 3D Drucken Anwendung findet. Dies macht die Weiterverarbeitung mit zusätzlicher Software, wie MeshLab, überflüssig.

Ich diskutiere die Herausforderungen mit denen ich während der Konstruktion der verschiedenen Prototypen konfrontiert war, welche Lösungen ich gefunden habe und am Ende wird die Evaluierung des finalen Prototypen vorgestellt.

Schlussendlich werdem dem Leser eine ausführliche Baupläne zum Nachbau des kompletten Systems für unter 130€ bereit gestellt.

Acknowledgements

At this place, I would like to thank Prof. Dr. Jan Borchers and Prof. Dr.-Ing. Kowalewski for giving me the possibility to do this work. Special thanks go to my supervisor René Bohne and to Andre Stollenwerk for their constant support and useful input. I also would like to thank all the people from the FabLab Aachen for their constructive critique and all the people that took part in the user studies. Special thanks also to Mihir Joshi for checking the language and spelling of this thesis.

Thank you!

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in Indian English.

Download links are set off in colored boxes.

File: myFile[a]

[a] http://hci.rwth-aachen.de/tiki-download_wiki_attachment.php?file_number.file

Chapter 1

Introduction

"The next phase of the digital revolution will go beyond personal computation to personal fabrication."

—Neil Gershenfeld

The digital revolution will be most successful when people start organizing themselves into working groups [MLGG02]. This idea is realized by FabLabs all around the world: Each FabLab consists of a collection of working tools like laser-cutters, 3d-printers, milling machines etc. which allow to design and prototype tools for a wide range of application fields. A FabLab is an open laboratory, which means it is not exclusively used by academics. Every amateur can use the equipment. Hence the aspect of usability is of great importance.

The community can us these FabLabs to start "creating their own technological tools for finding solutions to their own problems."[MLGG02] Right now, the problem is the following: people have to use highly professional modeling software to create a digital representation of their object. Whereas sometimes it is way easier to use simple and known tools (plaster for instance) to model an object. However, this model needs to be digitalized so it can be further processed on the computer or even be shared among the community.

One solution to this problem are 3D scanners. A 3D scanner is a device that creates a digital model of a real world object by analyzing its shape and color. The field of application comprises computer graphics, robotics, industrial design, medical diagnosis, cultural heritage, multimedia, entertainment, as well as rapid prototyping and computer-aided quality control. [WMMW06] [LT09]

There is a variety of approaches to perform 3D scanning, including structured light, coded light, time of flight etc. [Bla04] The approach of choice for simple and precise 3D scanning is triangulation-based laser range scanning which consists of calculating the intersection of the illuminating laser beam and the rays projected back to the camera. [WMMW06] This technology is well-known for more than twenty years [HMK82] [PM83]. In the last couple of decades, lasers and cameras have become inexpensive. 3D scanners can now be built with off-the-shelf parts. [LT09]

Although a few non-professional 3D laser scanners already exist, I will show in section 3.1—"Design and Requirements" that there is much room for improvement, especially on the software and usability side. I propose such a low-cost system for 3D data acquisition with integrated surface-mesh generation and support for ready-to-print models. The provided software minimizes the post-processing task which is cumbersome for most novice users. I will show that it is possible for untrained users to produce a replica of an object with only a few clicks.

1.1 Thesis Overview

The thesis is organized as follows. In Chapter 2, I provide an overview off different 3D scanning approaches. A set of 3D scanners is presented and the math behind laser range scanners is briefly explained. I give a few words on camera calibration and point-cloud to surface-mesh transformations.
Chapter 3 starts by listing the requirements for the project. Then I present my soft- and hardware prototypes. Each prototype is accompanied by a short discussion.

In Chapter 4, the final prototype is evaluated. I show that all the specified requirements were met. Both the usability of the complete setup and the precision of the scanner are analyzed.
The thesis concludes with a summary in chapter 5, followed by a quick look onto future work.

Chapter 2

Related work

This chapter starts by giving an overview of different 3D scanning approaches, then I throw some light on a number of available scanners. Later, the focus is put on camera calibration and depth reconstruction from triangulation. Finally, the generation of surface-meshes from point-clouds is discussed.

2.1 3D Scanning Approaches

In this section, I will present a set of different 3D scanning approaches. 3D scanning can be sub-divided into two main categories: contact based and non-contact based.

2.1.1 Contact Based

Contact based scanners are slow and may harm the object.

Contact based scanners are in direct contact with the surface of the object to be scanned. A probe is used to estimate the shape by recording the displacement of the probe as it slides across the solid surface. While effective, such contact-based methods can alter or even harm fragile objects. Besides, they require long periods of time to build an accurate 3D model. [LT09]

2.1.2 Non-contact Based

Non-contact based scanners are fast and avoid damaging the object.

Non-contact based scanners can further be sub-divided into passive and active scanners. While both scanners rely on one or multiple light sources to reconstruct the shape of the object, passive scanners do not need to actively control the illumination source, instead they rely entirely on ambient light. Scanning without contact allows high speed scanning while avoiding physically damaging the object, which is important for instance while replicating antiques. [GSGC08]

Passive scanning

Stereoscopic imaging and the correspondence problem

The most famous representative of this technique is probably *stereoscopic imaging*. In this approach, two cameras are used to mimic the human visual eye system. Knowing the position of the cameras it is possible for each camera system to independently establish an equation of a camera-ray (see section 2.3.2—"Basics of Triangulation") corresponding to a point in space. By calculating the intersection of those resulting two camera-rays, the position of the specified point can be extracted. This concept can even be extended by adding more cameras. However, all such passive triangulation methods require correspondences to be found among the various camera views, which remains an open and challenging problem [SRDD06], especially when scanning flat or periodic textures. This problematic is known as the *correspondence problem* .

Shape-from-silhouette and shape-from-focus

Many alternative passive methods emerged, avoiding the correspondence problem. [A94] proposes a *shape-from-silhouette* algorithm. Another technique, *shape-from-focus*, uses the focus of the camera to recover depth. Only objects close to the plane of focus will appear in sharp contrast, whereas distant objects are blurred. [NN94] [WN98]

Active Scanning

Sensitive to surface material properties

Active optical scanners avoid the correspondence problem by replacing one of the two cameras from stereoscopic imaging with a controlled illumination source, such as a laser or a projector. In comparison to non-contact and passive methods, active illumination is often more sensitive to surface material properties like reflection and absorption. [LT09]

Laser range scanners and structured light

Using a line laser, *laser range scanners* create a planar sheet of light, which defines a plane in space. This plane of light is then mechanically swept across the surface of the object to be scanned. The depth is perceived by calculating the intersection of this plane with the set of lines passing through the laser reflection on the surface and the camera's center of projection. See section 2.3.2—"Basics of Triangulation" for further explanations and illustrations. A digital *structured light* projector can be used to eliminate the mechanical motion required to translate the laser stripe across the surface. The projector can display different patterns like multiple lines at one time, which diminishes the scanning time [SPSB04].

Problems: occluded regions and dynamic scenes

While effective, laser range scanners and structured light scanner remain difficult to use if moving objects are present in the scene. Due to the separation of the light source and the camera, certain occluded regions cannot be recovered. This limitation require multiple scans to be merged, further increasing the data acquisition time. Both laser range scanners and structured lighting are ill-suited for scanning dynamic scenes.

Time-of-flight rangefinders

Another approach are *time-of-flight rangefinders*. They estimate the distance to a surface from a single center of projection, so occluded regions do not appear. Since the speed of light is known, it is possible to estimate the depth by measuring the elapsed time between emitting and receiving a pulse of light. However, the depth resolution and accuracy of such systems remain below that of laser range scanners and structured lighting. [LT09]

2.2 Comparing Available 3D Scanners

Among all the presented scanning technologies, line-laser range scanning offers the best combination of accuracy, affordability, scanning speed, and simplicity. This technology is also used by professional scanners such as the Artec 3D Scanners[1] and the NextEngine[2] , whose prices start at 10.900 € and 2.995 $ respectively. In this section I present a list of more affordable 3D scanners.

2.2.1 David Scanner

Figure 2.1: David Scanner

A starter kit of the David Scanner[3] is available for 399 €. 360° scans are possible, although the user needs to turn the object manually between multiple scans. The software is proprietary and runs only on Windows. Minimal equipment is needed, such as a line laser, a camera and a calibration pattern which can be printed out. Using the DAVID-Shapefusion software the user can manually merge multiple point-clouds from different scans and convert them to a surface mesh by-hand.

[1] http://www.artec3D.com/
[2] http://www.nextengine.com/
[3] http://www.david-laserscanner.com/

2.2.2 MakerBot 3D Scanner

Figure 2.2: Maker Bot 3D Scanner

The MakerBot 3D Scanner[4] (formerly know as *Cyclops*) is a 3D scanning mounting kit for a pico projector, a webcam and an iPhone or iPod. The mounting kit is available for 50$. A set of additional open-source software is required (ThreePhase, PeasyCam, ControlP5, Processing). Extensive instructions are provided on the website to perform a single face scan. Using Blender and MeshLab the point-clouds are transformed into surface meshes which can then be printed on the MakerBot.

2.2.3 3D Photography on your Desk

This project requires very little hardware: a camera, a desk-lamp, a pencil and a checkerboard.[BP98] The checker-board is used for calibration. The pencil is waved between the lamp and the object to be scanned, casting a shadow on the object. The 3D shape of the object is extracted from the spatial and temporal location of the observed shadow. Several implementations of this technique are available on the project's website[5] .

[4]http://wiki.makerbot.com/3D-scanner
[5]http://www.vision.caltech.edu/bouguetj/ICCV98/

Figure 2.3: 3D Photography on your desk

2.2.4 Maker Scanner

Figure 2.4: Maker Scanner

The MakerScanner[6] software is open-source. Only single face scans are possible. The scanner setup can hold a webcam and a laser. The laser needs to be rotated manually. The parts for the scanning setup can be 3D-printed and are available on thingiverse[7] .

2.2.5 Microsoft Kinect

The Kinect is not suited for 3D scanning

In 2011, Microsoft released the Kinect, a device for recognizing and tracking player identity[LMW+11]. Besides a set of other sensors, the Kinect also includes infrared projectors and cameras which are used for depth detection, hence it could also be used to build a 3D scanner. However, according to [Rog11], the depth perception technology in the Kinect works best at distances of 6-8 feet ($\approx$ 2m). Thus, the Kinect is better suited for identifying "likely humans in the scene and the likely positions of their arms and legs."[Rog11] than for precise short-range 3D scanning.

2.3 Camera Calibration and Triangulation

In the previous section, a set of 3D scanners was introduced. In the next section, the thesis focuses on the theory behind 3D laser range scanning. First, the pinhole camera model is introduced along with explanation about camera calibration using functions provided by the *Open Computer Vision Library (OpenCV)* [BK08], then the basics of triangulation (used to recover the depth) are explained.

2.3.1 The pinhole camera model and camera calibration using OpenCV

In this model, exactly one light ray enters the camera through a pinhole from every point in the scene. The light is "projected" onto an image plane, behind the pinhole. To simplify the calculations, the pinhole camera is now rearranged: the image plane is put in front of the pinhole which is now reinterpreted as the center of projection. The distance between the center of projection and the image plane is known as the focal length f. Figure 2.5 shows the rearranged pinhole camera model setup.

[6] http://wiki.makerbot.com/makerscanner
[7] www.thingiverse.com

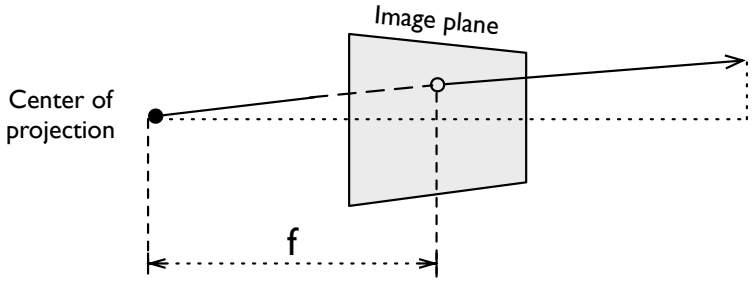

Figure 2.5: The pinhole camera model

Intrinsic properties

Notice that there are actually two focal length parameters f_x and f_y. The reason for this is that the individual pixels on a typical low-cost camera are rectangular rather than square. Furthermore, the center of the chip is usually not aligned with the optical axis. Thus, two new parameters, c_x and c_y, are introduced to model a possible displacement (away from the optic axis) of the projection screen.

These parameters f_x, f_y, c_x and c_y are known as the *intrinsic* properties.

Lens distortions

In theory, a lens can be defined without distortions. In practice, however, no lens is perfect. There are two non-negligible types of distortion: Radial distortions ("fish-eye" effect) arise as a result of the shape of lens, whereas tangential distortions arise from the lens not being exactly parallel to the image plane. (Compare chapter 3 of [LT09]). Radial distortions are represented by at least three parameters k_1, k_2 and k_3. Even more parameters can be introduced for cameras with higher radial distortion. Tangential distortion is characterized by two additional parameters, p_1 and p_2. All these parameters are explained in detail in chapter 11 of the OpenCV book [BK08].

Extrinsic properties

The parameters R and T, which are referred to as the *extrinsic* parameters of the camera, describe the location and orientation of the camera in world coordinates. In three-dimensional space each of them can be represented as a three-dimensional vector, which gives us a total of 6 extrinsic parameters.

The OpenCV method $cvCalibrateCamera2()$, explained on page 398 of the OpenCV book, can be used to calculate the intrinsics, the distortion and the extrinsic parameters. As input, the function uses pictures of several chessboard patterns as explained in [Zha00]. OpenCV's approach to calculate the intrinsics is derived from [HS97]. Both [XRL11] and [WLZ10] deal with camera calibration techniques based on OpenCV.

2.3.2 Basics of Triangulation

In this section, the concept of triangulation is explained. Giving a complete and detailed insight into this topic would go beyond the scope of this thesis. Therefore, I only introduce the basic concepts of triangulation. For the interested reader, I suggest reading chapter 2 *The Mathematics of Triangulation* of [LT09].

Every piece of hardware used in the scanner is represented by a model in the software. For the camera, the *Camera Pinhole model* is used as explained in section 2.3.1—"The pinhole camera model and camera calibration using OpenCV". Figure 2.6 shows the basic model used for triangulation.

The line laser can be modeled as a simple plane in space, assuming the opening angle of the laser lens is large enough and the laser is placed sufficiently far away from the object.

The intersection of the laser light plane with the object being scanned generally contains many illuminated curved segments (see Figure 2.6 red curved line). These segments are composed of many illuminated points. A single illu-

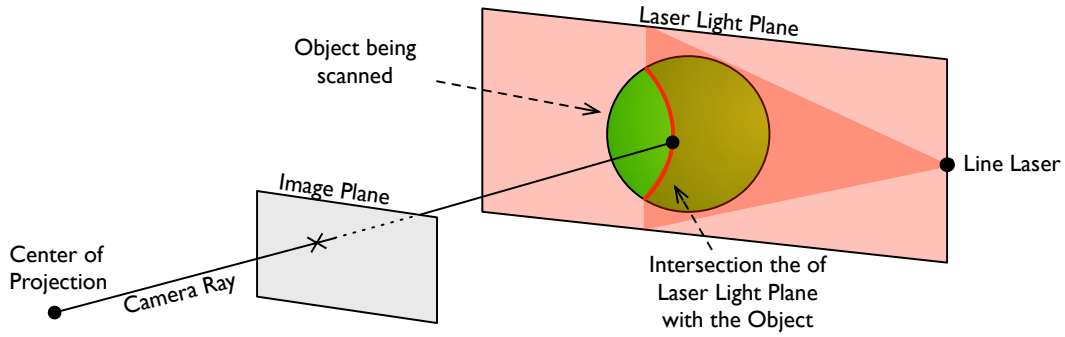

Figure 2.6: Model of 3D surface recovery using ray-plane-intersection triangulation.

minated point, visible to the camera, defines a *camera ray*. Here, it is assumed that the positions and rotations of the laser and the camera are known with respect to the global coordinate system. In the previous section it is explained how to estimate these values for the camera. Under this assumption, the equations of the laser light plane, as well as the equations of camera rays corresponding to illuminated points, can be calculated. The location of the illuminated points can be recovered by intersecting the laser light plane with the camera rays corresponding to the illuminated points. By rotating the object (and the laser) during the scan, all sides of the object are illuminated, allowing recovery of a 360° surface model. [LT09]

2.4 Surfaces from Point-Clouds

As soon as a point-cloud is acquired, it enters the post-processing pipeline whose eventual goal is to produce a polygon surface mesh, which can then be used for 3D printing.

Powercrust

The *powercrust* algorithm [AC01] can be used for this task: "The power crust is a construction which takes a sample of points from the surface of a three-dimensional object and produces a surface mesh." The algorithm delivers a watertight surface-mesh and eliminates the polygonization, hole-filling or manifold extraction post-processing steps re-

quired in other surface reconstruction algorithms, which is ideal for our needs. Besides providing an implementation of the algorithm, the authors also prove "good empirical results on inputs including models with sharp corners, sparse and unevenly distributed point samples, holes, and noise, both natural and synthetic".

Figure 2.7: "Laser range data, the reconstructed watertight polygonal model, and its simplified medial axis." [AC01]

Point Cloud Library and OpenNI

Alternatively, one can refer to the *Point Cloud Library (PCL)* [RC11] presented in 2011. The library contains state-of-the art algorithms for filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. Furthermore, it offers native support for OpenNI [Ope11] 3D interfaces, and can thus acquire and process data from devices supporting this standard, such as the Microsoft Kinect.

Chapter 3

Own work

The previous chapter introduced the concept of 3D scanning. A number of DIY 3D scanners were presented. In the following chapter I will present the design, the software as well as the hardware prototypes of the FabScan 3D scanner.

3.1 Design and Requirements

This project rooted in the idea to supply the FabLab Aachen[1] with a low-cost 3D scanner. In an initial user study, I surveyed people visiting the FabLab about 3D scanners. The following requirements of the scanner were identified:

- **Affordability** The scanner should be affordable, to an extend that amateurs with a modest budget can purchase it.
- **Ready-To-Print** The scanning software should make it easy for the user to produce a watertight ready-to-print surface meshes of the scanned object. Thus, eliminating, or at least minimizing, the cumbersome post-processing pipeline.

[1] http://fablab.rwth-aachen.de

- **Do-It-Yourself** Anyone should be able to assemble the scanner with access to a laser-cutter, 3D printer, soldering iron, etc. The parts of the scanner could even be provided as an inexpensive construction kit.
- **360°-Scan** The scanner should be able to automatically scan the object from all viewing angles in only one scan, without requiring the user to manually rotate the object.
- **Portability** The software should be portable to multiple platforms or at least run on a Mac as the FabLab Aachen is based on Macs.
- **Usability** Untrained users should be able to cope with the system. Ideally, the scanning process should be fully automatic. The user should only need to press a button once to start a scan.

3.1.1 Comparing Scanners

The following table summaries the scanners presented in chapter 2. They are compared to the previously established requirements.

	Affordability	*Ready-To-Print*	*Do-It-Yourself*	*360°-Scan*	*Portability*	*Usability*
DAVID Scanner	✔	✔	✔	✔	✘	✔
MakerBot 3D Scanner	✔	✘	✔	✘	✔	✘
3D Photography on your desk	✔	✘	✔	✘	✔	✘
Maker Scanner	✔	✘	✔	✘	✔	✘

Table 3.1: Summary of the compared scanners

It should be noted that "usability is not a quality that exists in any real or absolute sense" [Bro11]. Here usability is simply defined wether it is possible to perform a scan using a very reduced number of clicks or not.

Figure 3.1: Physical objects used for prototype testing. From left to right: the bear, the skull, the duck and the cup.

3.2 Hardware Prototypes

The final hardware prototype is the result of several DIA-cycle iterations (Design Implement Analyze) [Car92] [Bor01]. In this section, I present all the prototype iterations. I show some of the scanning results of each prototype and shortly discuss what I learned from the prototype and how I used this information to improve the next prototype. Throughout the prototyping phase, I experimented with several objects of different size, material, color and shape. In the following, I will refer to those objects with the names as presented in the caption of figure 3.1.

3.2.1 First Hardware Prototype

The first prototype is based on the Maker-Scanner, presented in section 2.2.4—"Maker Scanner". The scanning setup is composed of four primary items: a Logitech Quickcam Pro 9000 webcam, a line laser on a servo motor, an Arduino Uno[Uno11] and two planar surfaces. Note that the laser and webcam must be separated so that the laser light plane and camera rays do not meet at small incidence angles, otherwise triangulation will result in large errors. [LT09] On the other side, the angle must not be too large to avoid obscured areas on the surface of the object. The two planes should form a right angle. The setup differs in one major point form the Maker-Scanner: whereas the laser of the Maker scanner is swept manually by the user,

this prototype incorporates a servo motor which takes over the role of the user moving the laser. This allows a fully automatic scan, hence the user is not involved in the actual scanning process. This point is important as the user should not need to interact with the system during a scan. The servo motor is controlled using an Arduino, which is connected to the computer over a serial port. The Arduino is also used to turn the laser on and off. This is used to extract a hight contrast image of the laser line (compare with section 3.5.1—"Laser Threshold"). The setup of the first prototype is shown in figure 3.2.

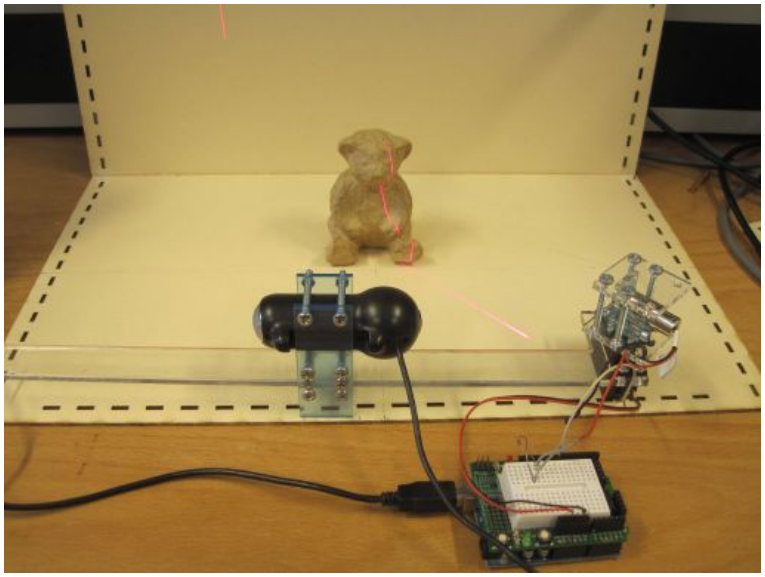

Figure 3.2: Setup of the first prototype.

Discussion

The first prototype showed that the triangulation principle gives acceptable results, even when using a rather simple setup. The result of a scan is shown in figure 3.3. One vertical line corresponds to one laser position. The relatively large gap between the different lines is due to the used servo: the minimal resolution of the servo is 1° per step. Also, it is not possible to get a full 360° scan of the object.

Figure 3.3: Scanning results from first prototype. Left: the physical object Bear. Right: the point-cloud of the Bear

The ambient light has a great influence on the scanning results. Ambient illumination must be reduced so that the laser line is clearly visible to the camera. However, it should also not be too dark to allow the camera to operate with minimal gain. Otherwise sensor noise will corrupt the reconstruction. The mentioned ”3D Photography on your desk” scanner has the same problem [BP98].

3.2.2 Second Hardware Prototype

To get a full 360° scan of the object the laser is no longer swept over the object but instead the object is placed on a turntable. The turntable is mounted on a stepper motor which allows it to turn. The stepper motor can make steps as small as 0.1125° (see also 3.4.2—“Selecting the scanning resolution”), which is largely sufficient for our purpose. The servo is no longer used during the scanning process, it only serves as a nice holder for the laser. Also I created a box for the scanner. The Arduino is now mounted inside the box and all the cables can easily be plugged into the front of the box. The scanner is now more portable and less time is needed to set it up. Later, I also created a cover which can be placed on top of the scanner to isolate it from ambient illumination. The setup is depicted in figure 3.4.

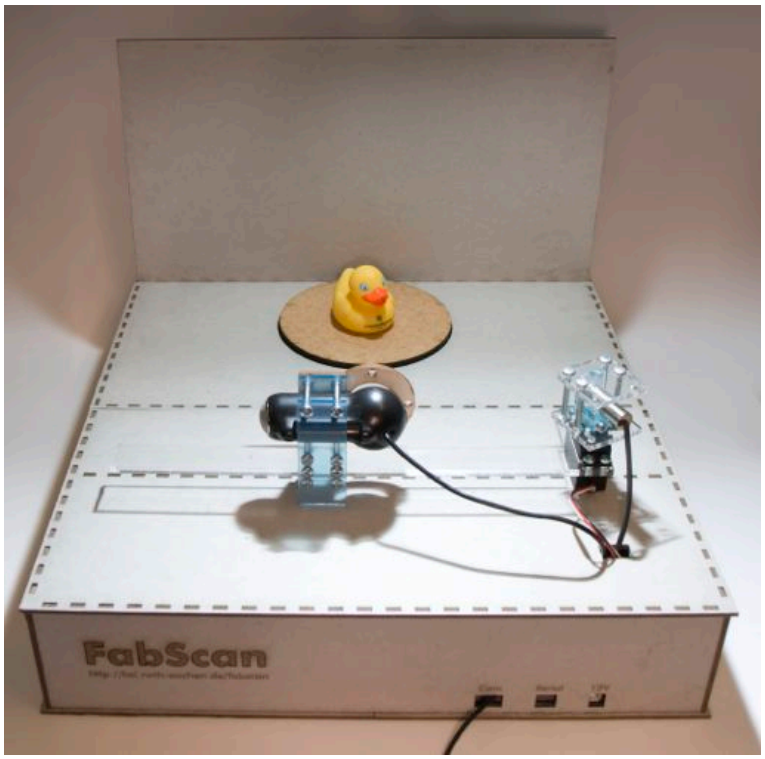

Figure 3.4: Setup of the second prototype. Cover not shown.

Discussion

With the second prototype much denser point clouds are achieved. However, there are still holes in the surface. See figure 3.5. There are multiple causes for these holes. Since the laser is no longer swiped over the object another problem became apparent: depending on the objects shape, some parts of the surface (marked with ① in the figure) are never illuminated by the laser. Thus, these parts will appear as holes in the point-cloud. Another notable problem is given by the color of the object. The color of the object has an impact on the scanning results. Some colors absorb the laser light, like the blue dots on the cup or the label on the front of the duck ②. For my objects the holes were sufficiently small enough, so the powercrust algorithm identified them as artificial holes and closed them. Number ③ on the cup shows a small displacement. As the

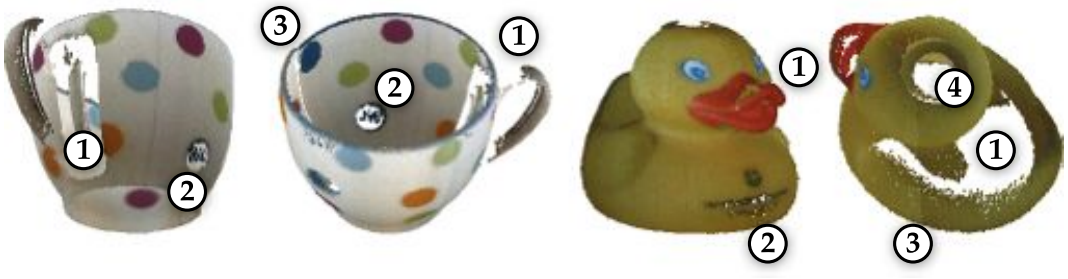

Figure 3.5: Point clouds generated with the second prototype from various point of views. Left: the cup. Right: the duck

material of the cup is slippery, it moved a little during the scanning process every time the turntable advanced a step. This can be solved by placing a round piece of black fabric onto the turntable which augments the grip. In addition, the black color of the fabric absorbs unwanted reflections of the laser line on the turntable. The material of the duck is less slippery, so it did not move during the scan. Nevertheless, one can notice slightly different colors. This is due to a different illumination between the start and end of the scan. This problem is eliminated by putting a cover on top of the scanner and adding a light source under the scanner so the illumination conditions stay the same during a scan. One could think that the light source is unnecessary if colored point clouds are not needed. Yet putting the object in complete darkness, with the laser as the only light source, makes it difficult for the camera's built-in auto focus feature to take sharp pictures. The strange displacement marked with ④ near the top of the ducks head was due to a software bug, which is fixed now. On a more general note, with the current setup where the object is rotated on only one axe, it is not possible to scan the bottom of the objects, nor the inside of the cup or the top of the ducks head ④.

On the hardware side, the design of the box is not ideal: The Arduino is not easily accessible and the material of the box wears out when opening the box to often. The connectors are also not well placed. In most cases, the computer which controls the scanner is placed either on the left or the right of the scanner, so should be the connectors.

3.2.3 Third Hardware Prototype

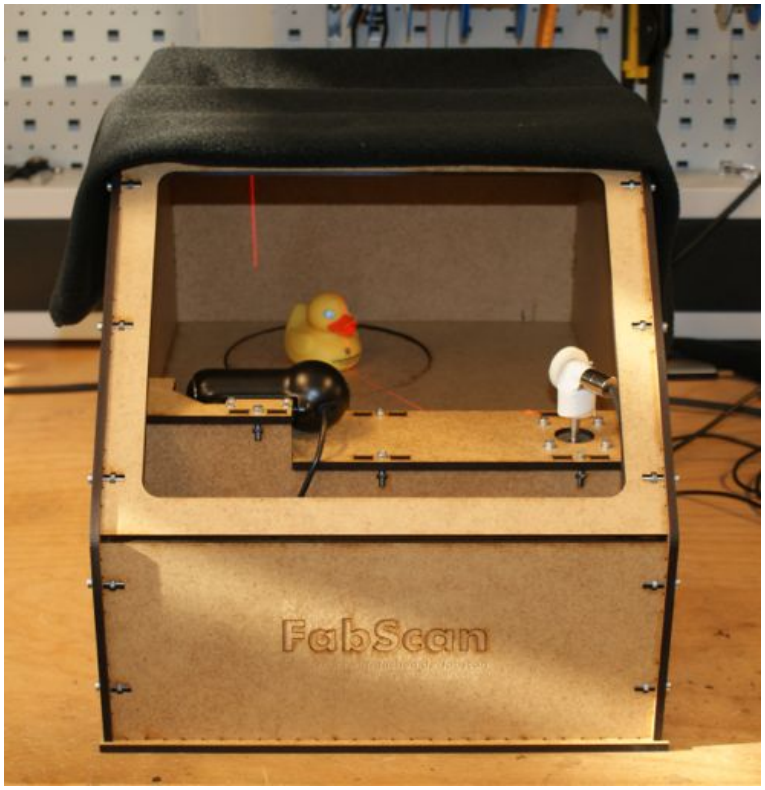

Figure 3.6: Setup of the third prototype

From the last prototype, I learned several points: The scanner should be easier to assemble and disassemble without destroying the material. The laser needs to rotate during a scan in order to cover more parts of the object surface. Since the servo motor is not precise enough, it should be replaced by a stepper motor. The construction by itself should allow to deduce the exact position of the laser relative to the turntable. Another, more stable and enduring material should be used.

The main idea behind the construction of this prototype is to fix all the critical parts, like the camera, the laser and the turntable on one single base plane. This single base plane can then be used as a very precise reference system which allows to deduce the exact positions of the turntable and the laser as those are both positioned on a stepper motor,

whose dimensions are also exactly known. The round and asymmetric shape of the Logitech camera makes it difficult to know the exact position just by construction. To solve this problem, software calibration can be used to determine the extrinsic values. OpenCV provides the function *cvFindExtrinsicCameraParams2* for exactly this purpose.

For this prototype, 5mm MDF (Medium-density fiberboard) is used. It is much more stable then the previously used gray cardboard. This allows a more precise construction. The downside of a more stable material is that it is less flexible, hence screws have to be used to make the different parts stick together. Also the connectors of the Arduino are now on the right side of the box. With the new construction, it is possible to quickly remove the cover, giving immediate access to the Arduino. The cover contains large holes which make it easy to put the object inside the scanner. To seal the inside of the scanner from ambient light, a black curtain is placed over the cover.

In this prototype, the laser is mounted on a stepper motor. The servo motor is completely removed. This allows precise movement of the laser during a scan. First, the laser line sweeps over the object, stopping at several positions where triangulation takes places. Then the turning table advances one step and the laser sweeps again over the object, this time in the opposite direction. This is repeated until the turning table made a full rotation.

Discussion

Scanning an object with multiple different laser positions minimizes the problem of obscured areas as explained previously with the cup and the duck. This produces denser point-clouds and potentially better surface-meshes. This is illustrated in figure 3.7.

Using a stepper motor instead of a servo motor also has a downside. The servo motor was controlled by specifying a position. The stepper motor is controlled by specifying a number of steps it should turn. This means that the current rotation of the laser is not immediately known. In this pro-

Figure 3.7: Surface meshes of the cup. The handle of the cup dramatically increases in quality when using more laser positions. Left: 1 single laser position. Middle: 3 different laser positions. Right: 5 different laser positions.

totype, it is assumed that the laser is pointing at a specific position (marked on the box) before every scan. If this is not the case, the user is required to manually turn the laser to that position. To overcome this cumbersome situation, one could either use additional hardware such as an end switch or a potentiometer, alternatively the position could be deduced using software calibration.

The idea to put to camera, the laser and the turntable on one single reference plane worked very well. Unfortunately, OpenCVs *cvFindExtrinsicCameraParams2* did not provide useful values for the extrinsic parameters. Using the older *OpenCV Framework 1.2* available from the RWTH IENT institute[2] improved the situation a little: although the *cvFindExtrinsicCameraParams2* still did not work as hoped, the function *cvCalibrateCamera2* at least delivered more realistic values for the extrinsics but still not precise enough. The reason for this is most probably a bug in either my software or the implementation of OpenCV. Since the position of the camera is a crucial part of triangulation and time was moving on, I decided not to use OpenCV but instead measure the extrinsic values physically. Obviously, this situation is not ideal.

The turntable is a critical part of the scanner. Its mechanical construction has a great influence on the scanning results. The turning table should be perfectly horizontal and turn in one single plane. Also the surface of the turntable should be ragged otherwise the object might slip when turning. This results in a distorted image as the software thinks the object has rotated but in fact it has not or only a little.

[2] http://www.ient.rwth-aachen.de/cms/opencv/

An important aspect is the time it takes for a scan to complete. As explained in the section 3.5.1—"Laser Threshold" two pictures are taken every time the turntables moves. The camera takes some time to focus when the image changes, so it takes some time until the image is stable. This time could dramatically be shortened when disabling some features on the camera like auto-exposure-time and auto-white-balance. Since the camera supports the UVC [For05] standard, this features can be turned off. I already have some working code, which does exactly this but it is not yet implemented in the this version of the prototype.

3.2.4 PCB - Arduino Shield

To control the two stepper motors, the laser and the LEDs I am using an Arduino with an additional self-made shield. The Arduino and both sides of the shield are shown in figure 3.8. The shield was manufactured on a CNC milling machine. It is simply plugged on top of the Arduino.

The maximum voltage for the laser is 4,5V. A diode is used to go from the 5V, provided by the Arduino, down to 4,3V. The laser is connected to ①, as shown in figure 3.8. The LEDs are connected to ②. The power supply for the LEDs is controlled with a transistor ③. The transistor is attached to a PWM capable pin on the Arduino so it is possible to precisely set the brightness of the LEDs.

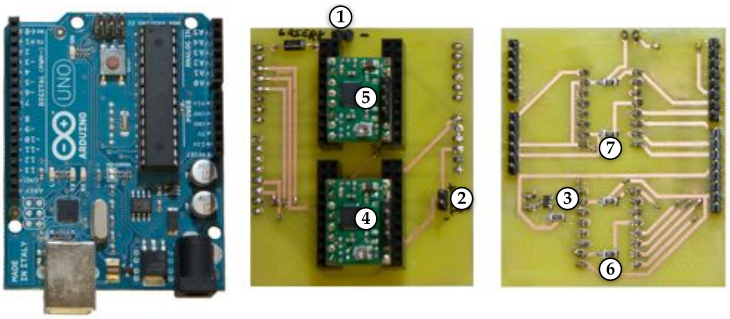

Figure 3.8: Left: Arduino Uno. Middle: Front side of shield. Right: Back side of shield

Number ④ and ⑤ show the Pololu A4983[3] stepper motor drivers. The stepper motor for the turntable is connected to ⑤ while the one rotating the laser is connected to ④. Without further modification, the stepper motors do 200 steps per 360°. By enabling micro-stepping, it is possible to achieve 3200 steps per 360°, which allows a precision of 0.1125°. The resistors ⑥ and ⑦ are used to enable micro-stepping for the laser stepper motor and the turntable stepper motor respectively.

3.3 Software

This section starts by introducing the protocol to control the FabScan. Then the implementation of the triangulation is explained as well as the needed transformation for the STL file format. The different supported point-cloud file formats are explained. Finally, we take a closer look at the development of the GUI and its functionality.

3.3.1 Communication Protocol

The protocol used to control the FabScan is quite simple. There are two types of frames. The first type is only one byte long, the second contains two bytes. The one-byte frame is used for simple control actions like turning on the laser or disabling a stepper motor. The 2-byte field contains an additional value byte. This value byte can be used to set the brightness of the light or make the selected stepper motor perform a given number of steps. All the possible combinations are shown in figure 3.9. When sending an action to a stepper motors, it must be made sure the correct stepper is selected. Stepper motors can be selected with the "Select Stepper [212]" byte followed by the stepper ID. The ID of the turntable stepper is 0, the one of the laser is 1.

The values for the control byte start at 200 for "historic" reasons. In the first prototype, all the values below 200 were used to set the angle of the stepper motor. At this point,

[3] http://www.pololu.com/catalog/product/1201/resources

Figure 3.9: FabScan Communication Protocol. The value or the range for each byte is specified in brackets. Left and Middle: single byte frames. Right: dual byte frames

since the servo motor is no longer used, they could just as well start at 0.

To make sure that there is actually a FabScan connected to the selected serial port, and not some other device which happens to have a serial port running, a "FabScan Ping [210]" message is sent, which needs to be answered with a "FabScan Pong [211]". When answered correctly, it is clear that a FabScan is connected to that serial port.

The computer talks to the FabScan over serial communication. On the Arduino side, the standard *Serial Library* is used. On the Mac, the class FSSerial is responsible for serial communication. It is based on Apple's IOKit and the example provided on the Arduino page[4]

3.3.2 Implementing Triangulation

For 3D depth recovery, I use line-ray triangulation as explained in section 2.3.2—"Basics of Triangulation". Due to the known geometry of the hardware it is possible to simplify the model without losing to much precision. The following assumptions are made:

- The laser line is perfectly vertical.
- The camera is pointed perfectly straight and perpendicular to the back of the box.

[4] http://www.arduino.cc/playground/Interfacing/Cocoa

The error introduced by these assumptions is relatively small and it simplifies the math. The position of the laser and the camera are known by construction. Also the position of the laser line hitting the back of the box is known, since the rotation and position of the laser are known. Now the laser line is extracted from the camera picture as explained in section 3.5.1—"Laser Threshold". The positions of the reflected points on the surface of the object are transformed from the image plane coordinate system into the world coordinate system. By making use of the previous assumptions, all these points can be projected onto one horizontal plane, allowing us to move from 3D into 2D. This situation is depicted in 3.10. This simplification implies the loss of the hight information, but it can be reintroduced as we will see later. We simplified the 3D line-plane intersec-

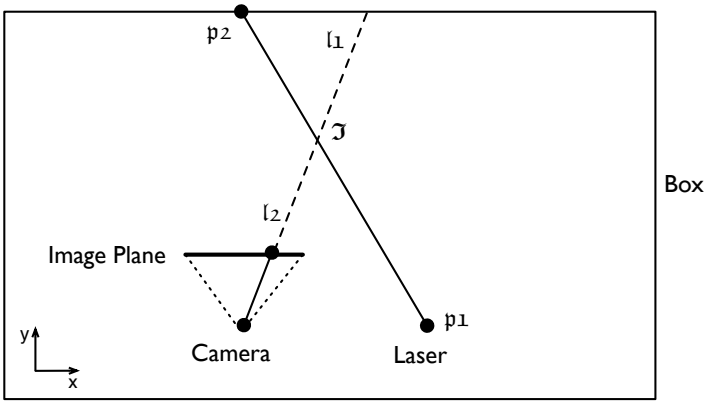

Figure 3.10: 2D line-line intersection

tion problem to the easier 2D line-line intersection problem.

Using two known points p_1 and p_2, we can establish the equation of a line l_1. Generally, the equation of a 2D line is given by:

$$y = a \cdot x + b \tag{3.1}$$

Points are represented as a tuple $p = (x_p, y_p)$. Specifically, the equation for l_1 is given by:

$$l_1 \equiv y = a_1 \cdot x + b_1 \tag{3.2}$$

where a_1 can be expressed depending on p_1 and p_2:

$$a_1 = \frac{\Delta y}{\Delta x} = \frac{y_{p2} - y_{p1}}{x_{p2} - x_{p1}} \tag{3.3}$$

Inserting p_1 (or p_2) into (3.2) and solving for b_1 gives us:

$$b_1 = y_{p1} - a_1 \cdot x_{p1} \tag{3.4}$$

At this point we have the complete equation for l_1. We use the same procedure to calculate l_2. Knowing the equations of two lines l_1 and l_2 we can calculate their intersection $\mathfrak{I}$.

$$l_1 = l_2 \tag{3.5}$$
$$\Leftrightarrow a_1 \cdot x_{\mathfrak{I}} + b_1 = a_2 \cdot x_{\mathfrak{I}} + b_2 \tag{3.6}$$
$$\Leftrightarrow x_{\mathfrak{I}} = \frac{b_2 - b_1}{a_1 - a_2} \tag{3.7}$$
$$\text{with } a_1 \neq a_2 \tag{3.8}$$

Again, $y_{\mathfrak{I}}$ can be calculated inserting $\mathfrak{I}$ into (3.2):

$$y_{\mathfrak{I}} = a_1 \cdot x_{\mathfrak{I}} + b_1 \tag{3.9}$$

Now the depth in real world coordinated of the scanned point is known, hence we can simply scale the height which is in image plane coordinates to get the world coordinates which gives us the 3D representation of the scanned point. These calculations are repeated for every point on the reflected laser line on the surface of the object. This set of points still needs to be rotated using affine transformations in order to take the rotation of the turntable into account.

3.3.3 From Point-Clouds to Printable Files

As soon as a point-cloud is acquired, it enters post processing whose eventual goal is to produce a polygon surface-mesh which can be used for 3D printing. Transforming the point-cloud into a polygon surface mesh is done using the powercrust algorithm as presented in section 2.4—"Surfaces from Point-Clouds".

We want to export the model into STL which is a stereolithography file format used among others for 3D printing. The format is shortly defined below:

Definition:
STL File Format

STL FILE FORMAT:
"An STL file is a triangular representation of a 3D surface geometry. [...] The native STL format has to fulfill the following specifications: (i) The normal and each vertex of every facet are specified by three coordinates each, so there is a total of 12 numbers stored for each facet. (ii) Each facet is part of the boundary between the interior and the exterior of the object. The orientation of the facets (which way is "out" and which way is "in") is specified redundantly in two ways which must be consistent. First, the direction of the normal is outward. Second, the vertices are listed in counterclockwise order when looking at the object from the outside (right-hand rule). (iii) Each triangle must share two vertices with each of its adjacent triangles. This is known as vertex-to-vertex rule. (iv) The object represented must be located in the all-positive octant (all vertex coordinates must be positive). " [FF11]

The powercrust faces are not triangles. Thus, the polygons need to be transformed into triangles and the normals need to be calculated. This is done by taking the first three vertices of every face and combining them to a new face. This procedure is repeated for all the remaining vertices. However, this approach can create faces with small angles. Small angles add errors when trigonometric functions are applied on them.

The normal n of a flat face with the vertices v_1, v_2 and v_3 is defined by the cross product:

$$n = (v_1 - v_2) \times (v_2 - v_3) \tag{3.10}$$

The normal n still needs to be normalized by dividing every component of the vector with the vectors length l:

$$l = \sqrt{x_n^2 + y_n^2 + z_n^2} \tag{3.11}$$

Now all the data for the STL file is known.

A ASCII STL file starts with `solid` and ends with `endsolid`. In between, all the faces are listed, as shown below:

```
facet normal n_x n_y n_z
outer loop
vertex v_1x v_1y v_1z
vertex v_2x v_2y v_2z
vertex v_3x v_3y v_3z
endloop
endfacet
```

3.3.4 Point-Cloud File Formats

Besides exporting the surface-mesh of the scanned model, it is also possible to export the point-cloud itself. The software supports three different point-cloud file formats: PTS, PLY and PCD.

PTS

The PTS file format is very basic. Every line corresponds to one point in the cloud. Each line contains only the position of the point, it has the format `x y z`. The file has no header nor footer.

PLY

The PLY file format is supported so the point-clouds can be opened in MeshLab. Although the format can be used to store polygons, here only the point-cloud part is supported. Similar to the PTS format, every line corresponds to a point. In addition to the position, a line also contains the color information. The format is `x y z r g b`. The type for each field is defined in the header. A full documentation of the file format is provided here[5] .

[5] http://paulbourke.net/dataformats/ply/

PCD

The Point Cloud Data (PCD) file format is part of PCL, see section 2.4—"Surfaces from Point-Clouds". PCD is also the native file format in FabScan, which means it is the default format for saving point-clouds. PCD files can also be opened with FabScan. A full documentation of the file format, along with an extensive motivation for yet another file format is provided at the PCL Documentation[6] .

3.4 GUI and Functionality

In this section, I shortly present the different GUI prototypes. Then the basic as well as the advanced functions of the software are explained.

3.4.1 User Interface Prototypes

The graphic user interface of FabScan is largely inspired by Apple's Photo Booth[7] as the concept of taking a picture or recording a video is somewhat similar to performing a 3D scan. See figure 3.12 for a comparison of PhotoBooth and the final user interface design.

The first user interface prototype was a paper prototype. Paper prototypes are quick and cheap to develop. They are not detailed, so the designer and the user can focus on important high-level user interface design development. Additionally, they can quickly be changed to reflect the users input.

The user studies with this paper prototype have shown that the concept of making it similar to a known software (PhotoBooth) helped to quickly understand the GUI elements. The idea of pressing the scan button in order to start a scan became clear instantly. Further studies have shown that the

[6] http://pointclouds.org/documentation/tutorials/pcd_file_format.php
[7] http://www.apple.com/macosx/apps/#photobooth

Figure 3.11: First paper prototype

Figure 3.12: FabScan compared to Photo Booth. Left: Photo Booth. Right: FabScan

panel on the bottom of the window showing previous scans is not really needed. Hence, it was elided from the following prototypes.

The final software prototype adds two interface elements: similar to the ”Effects” button in PhotoBooth, a popup-menu for setting the resolution is added. Changing the resolution has shown to be used quite often, so it should be easily accessible. Buttons for switching between different views are added on the bottom left of the window, see figure 3.12. This allows switching between the PointCloud view and the SurfaceMesh view which is a major part of the software functionality, see 3.3.2—“Implementing Triangulation”.

3.4.2 Basic Functionality and Advanced Settings

Performing a scan

To start a scan, the user simply presses the button in the middle. Then the scan automatically starts. The points are added to the view as they are scanned. This allows the user to immediately see when something is going wrong. Then scan can then be interrupted. After a successful scan, the software plays a beep sound.

Selecting view mode

The user can switch between *Point Cloud* view and *Surface Mesh* view. This is done by selecting one of the buttons on the lower left corner in the main window.

Selecting the scanning resolution

The user can chose between multiple scanning resolutions. This is done by selecting one of the options in the popup menu on the bottom left of the window. The scanning resolution influences the density of the points in the generated

point-cloud. A higher scanning resolution means more points but also a longer scanning time. When scanning a new object, it is recommended to start with a quick low resolution scan in order to verify if the resulting point-cloud is acceptable. Some objects depending on there material, size, surface and color may not be well suited for scanning. See also section 3.5—"Better Scanning Results".

The following table shows the meanings of the different resolution options.

Resolution	Degrees per step	microsteps
Lowest	18°	160
Low	9°	80
Normal	4,5°	40
High	2,25°	20
Higher	1,125°	10
Overkill	0,1125°	1

Table 3.2: The different available scanner resolutions

The "Overkill" options is the highest possible resolution. This upper limit is defined by the A4983[8] stepper motor driver from Pololu. The stepper motor has 200 steps per revolution. Using micro-stepping it is possible to rotate the motor one sixteenth step which equals 0,1125°.

3.5 Better Scanning Results

Using the user interface as explained previously, makes it possible to get acceptable scanning results with little more work then pressing a button.

Sometimes better models or more dense point clouds are needed. Getting better scanning results is possible by tweaking a few advanced parameters. These parameters can be found in the menu under "FabScan/Control Panel...". The parameters can be changed in the bottom

[8] http://www.pololu.com/catalog/product/1201/resources

part of the *Control Panel* window. In the following these parameters are explained.

Setting the stepper motors resolution

The first parameter defines how many micro-steps the turntable turns in each step. By setting a lower value, the point-cloud becomes more dense, but the scanning takes more time. The second parameter specifies how many samples should be taken from the scanned lines. This is useful to get nice quadratic polygons, depending on the first parameter.

The next parameters influence the laser stepper motor. The ”Laser steps” defines the number of different laser positions between each step of the turntable. The angle between the laser steps can be adjusted with the next parameter. A higher number of laser steps means less occluded areas on the objects surface, which leads to less artificial holes in the point-cloud. The scanning time is directly proportional to the number of laser steps.

3.5.1 Laser Threshold

For each step of the scanning process we extract the laser line. This is done by taking a picture when the laser is on and one when it is off. These color pictures are transformed into grayscale. Then the absolute difference between each pixel of the two pictures is calculated. If the difference is higher then the ”Laser Threshold” value, the pixel belongs to the laser line. This procedure produces a very high contrast line line.

You should change the threshold value, if the points on the scanned line are not aligned or when they differ too much in the depth. The threshold value should always be as high as possible. If it is too high, no points will appear during the scan.

Moreover, the laser line has a certain width. The right and the left limit of the laser line can be determined in the camera picture. The exact position of the laser line corresponds to the average value of those two lines. These limits are determined in pixels. When transforming them into world coordinates and then calculating the average value, it is possible to get sub-pixel accuracy. This is not yet implemented in the current software prototype.

3.5.2 Lower Limit

Sometimes points from the surface of the turntable are unintentionally scanned or the scanned object is positioned on a basement that should not be included in the model. Setting the ”Lower Limit” allows to elide all the scanned pixels below this value.

Chapter 4

Evaluation

In this section, I verify wether all the initial requirements as described in 3.1—"Design and Requirements" are met. Additionally the precision of the scanner is evaluated by scanning known reference objects and comparing them to the scanned results.

4.1 Affordability

In this paragraph we first calculate the price of the current prototype. The user studies, presented in 4.5—"User Studies", showed that most people think the price is appropriated. The following table shows a price listing [1] :

Component	Count	Price	Sum
Stepper motor	2	12,93 €	25,86 €
Pololu stepper motor driver	2	9,5 €	19,00 €
Arduino Uno	1	25,80 €	25,80 €
Red Line Laser	1	2,4 €	2,4 €
Logitech QuickCam Pro 9000	1	49,99 €	49,99 €
MDF + Screws + Nuts (approx.)			≈ 5 €
		Total	128,05 €

Table 4.1: Summary of the compared scanners

[1] all prices are from www.watterott.com if available

4.2 Ready-To-Print

One of the requirements was that the scanning software should provide the user with a watertight surface mesh of the scanned object that can be 3D printed without further post processing. To demonstrate the fulfillment of this requirement, the Bear was put into the scanner, he was scanned, transformed to a surface mesh and exported it as an STL file. All this happened in only 3 clicks. The 3D-printed model of the Bear is shown in 4.1 along with the original Bear. The point-cloud and the surface-mesh are also shown.

4.3 360° Scan

The turntable and the swiping laser in the final prototype make it possible to automatically scan the object from all viewing directions during one scan, without requiring the user to manually turn the object.

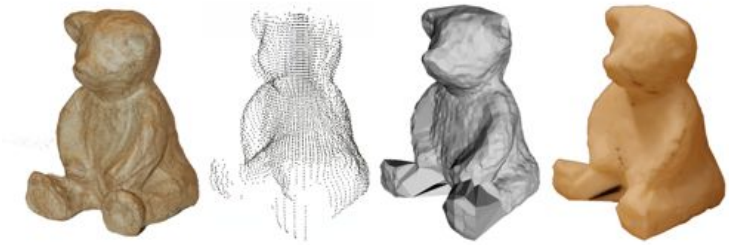

Figure 4.1: Replication process. From left to right: Original Bear, Point-Cloud, Surface-Mesh, 3D-printed Replica

4.4 Portability

The software was developed and tested on the Mac Platform running Snow Leopard. The FabScan software compiles and runs equally well on the latest Mac OS X Lion.

All of the software is written in platform independent C++ using OpenCV for image processing and OpenGL for displaying the point-clouds and surface-meshes, except for the user interface and serial communication where Objective-C is used. This allows to port the software to other platforms requiring only little additional work.

4.5 User Studies

It is difficult to verify the remaining requirements qualitatively. Therefore I did a formal user study to ensure that the system matches the users needs. I wanted to know if the system offers the right features, if it makes 3D scanning performable for untrained users. The tests also revealed unexpected or confusing situations. The evaluation took place in the FabLab which also happens to be the natural environment of the users. This makes the study more realistic although users might get easier distracted by noise and other interruptions.

At first, it was planned to do a separate user test for evaluating the user interface, but it turned out that the GUI contains only three elements, thus the user interface was tested along with the complete scanning setup.

In order to evaluate how the user reacts to the system, he is asked to accomplish several different task which cover the functionality of the system. These tasks are enumerated below:

- Task 1: Connect the scanner to the computer and start the software so we are ready to scan.
- Task 2: The user is given a object which should then be scanned.
- Task 3: The user should export the scanned model to an STL file.
- Task 4: Assemble the FabScan box. (Starting from the loose parts.)

The goal is to verify that the system is useable for untrained users. Hence, no usage instructions are given. If the user cannot accomplish a task, help is provided so he can finish.

The user tests were held in the FabLab Aachen, which is a natural environment for 3D Laser scanners. The users were between 20 and 25 years old, mostly students at the RWTH or visitors of DorkBot[2] .

The user tests showed that the all the participants had little to no troubles completing all of the assigned tasks. After the user test, the participants were asked to fill out the SUS [Bro11] questionnaire. (See A—"SUS - A quick and dirty usability scale"). The average results for all the participants are shown in the following table:

Question Nr.	Average Result
1	3,75
2	2,75
3	5,00
4	1,50
5	3,25
6	1,50
7	5,00
8	1,00
9	5,00
10	1,50

Table 4.2: Results of the SUS user questionnaire

SUS scores have a range of 0 to 100. A higher score means a better usability. The final SUS score is calculated as description in [Bro11]. The SUS score for this study is 84,375. According to [BKM08], this is in the range of "better products". Hence the usability and DIY requirement is fulfilled.

4.5.1 Suggestions And Feedback From The Users

The users were not always sure what the *Eye* symbol on the middle button should represent. When asked for possible alternatives, the answers were conflicting: some users liked

[2] www.dorkbot.de

the idea of a *Play* symbol as known from video players, others did not. A label *GO*on the button instead of a symbol has also been suggested. An appropriate *Start Scan* symbol still needs to be found.

When pressing the *Start Scan* button, a dialog could display short instructions like putting the curtain over the scanner. One user also found it strange that the scan process started immediately after pressing the button.

It was also suggested to create colored textures for the surface-meshes out of the colored point-clouds. The idea of a hardware version of the ”Start Scan” button was also mentioned.

4.6 Scanner Specifications

In this last section, some of the scanners technical specifications are presented.

4.6.1 Accuracy

The accuracy of the scanner is determined using two well known objects. These objects were scanned, then I compared the dimensions of the physical objects with the dimensions of the scanned objects.

The dimensions of the physical objects were determined with a sliding caliper. The digital models were measured in MeshLab[CR08] using the ”Measuring Tool”. It should be noted that it requires some practice to do precise measurements with the MeshLab measure tool, hence the measurements most probably contain a small error.

The first object is a truncated cone made of white polystyrene, the second object is the bear. As we 3D-printed the bear, we can simply use this replica and measuring it with the caliper as well, which is more pleasant then using MeshLab. At this point, it should be noted that the min-

imum resolution of the used 3D printer is 0,1778mm per print layer, which adds a small error to the measurements.

The results of the measurements for are shown in figure 4.2. For the bear, I also measured the distance from the nose to the tail, which is 101,6mm for the real bear and 100,7mm for the replica.

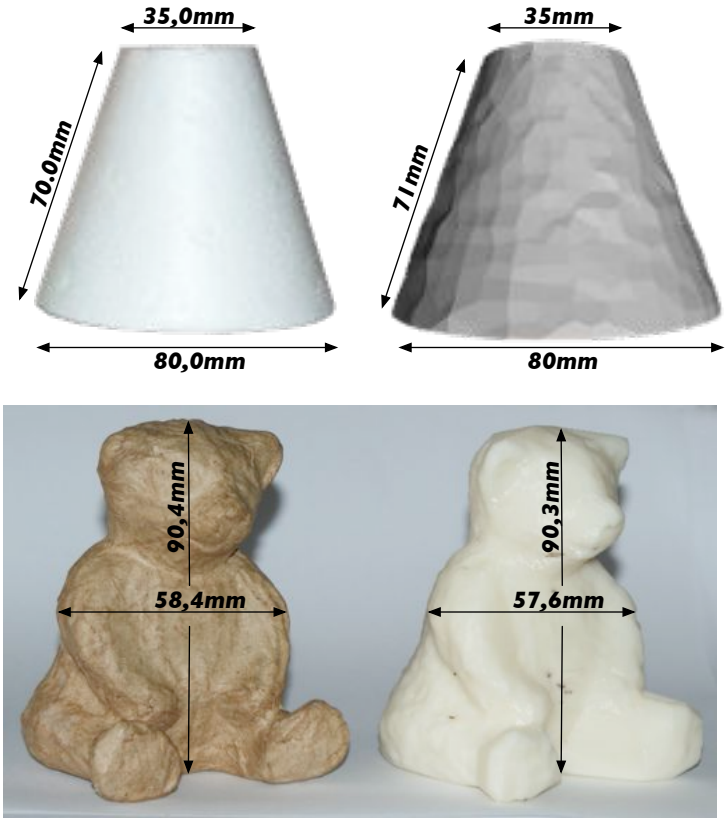

Figure 4.2: Real object (left) compared to replica (right).

4.6.2 Resolution

In this section we try to estimate the maximum resolution of the scanner. Here the resolution is defined as the minimum distance between two points in the point-cloud. Obviously, the resolution highly dependent on the current distance between the objects and the camera, the object and the laser and the position of the object on the turntable, which

makes it difficult to provide hard numbers. Additionally, it must be differentiated between the horizontal and the vertical resolution. The horizontal resolution is dependent on the step size of the laser and the turntable. The vertical resolution is not limited by the turntable or the laser, since the vertical laser line is continuous. Finally the camera has an impact on both resolutions.

The minimal step size of both stepper motors (laser and turntable) is 0.1225° (see 3.4.2—"Selecting the scanning resolution"). When only turning the turntable, the resolution near the middle of the table is close to 0mm, on the border the resolution is 0.149mm. When only turning the laser, the resolution is different for every step. When turning both the laser and the camera with their smallest possible step sizes, it becomes apparent that the resolution converges to 0mm or at least to a value not measurable anymore.

Hence, the limiting factor for the resolution of the scanner is defined by the resolution of the used camera. The camera image has a resolution of 1600×1200 pixels. Here it is assumed that the pixels of the camera are quadratic, so both horizontal and vertical resolution are equal. Clearly, the scanner resolution improves when moving the object closer to the camera. Thus, to measure the maximum resolution we chose the closest point of the turntable to the camera. At this point, 15cm correspond to 1500px. Hence 1px represents 0,1mm. However, at this distance the camera has problems to focus, which means a loss of precision. For comparison, at the opposite side of the turntable (point most far away from the camera) the resolution is 0.2mm and the image is very sharp.

Chapter 5

Summary and future work

In the previous chapters I presented the idea and described the development process of the FabLab 3D scanner. At last, I will give a summary of the most important aspects and the possible insights into the future.

5.1 Summary and Contributions

In this work, I created an 3D scanner for less then 150 €. An overview of different 3D scanning approaches was provided. The basic math needed for 3D scanning were explained. Extensive user studies allowed to define the requirements for the project which finally resulted from multiple prototypes. In the end, I gave a precise evaluation of the complete system.

My contribution to the community is a complete open 3D laser scanning construction kit available for download. A comprehensive and powerful software is provided to work along with the scanner. Since the communication protocol is open, the hardware scanner can serve as a base platform for other 3d scanning projects, hence eliding hardware building from scratch.

5.2 Future Work

In this chapter I will present ideas and suggestions which came up during the process of the thesis. As I concentrated on the basics of the FabScan 3D scanner, the following states potential further improvments.

5.2.1 Scheimpflug principle

One aspect of laser range scanners is that the sharpness of the picture changes along the projected laser line. When applying the *Scheimpflug principle* [LIAI11] by tilting the camera lens, it is possible to realize a constant image sharpness along the laser line.

5.2.2 3D Replicator

During the user studies it became clear, that the scanner is mostly used in combination with a 3d printer. Thus, it would make sense to merge both machines into one 3D replicator, possibly with one ”Copy” button.

5.2.3 MeshLab Plugin

MeshLab is an advanced mesh and point-cloud processing system for automatic and user assisted editing, cleaning etc. MeshLab’s modular architecture allows to add new functionality by writing new plugins [CR08]. A FabScan plugin for MeshLap would make it possible to scan immediately out of MeshLab.

5.2.4 Improve Hardware

The hardware could be pushed further by implementing more camera-laser instances which would allow to catch more of the obscured areas, such as the inside of the cup.

5.2.5 Intelligent Scanning

At this point the scanner is not aware of the objects appearance. It would be possible to built an intelligent scanner, that first checks the approximate object shape (compare shape-from-silhouette technique), then uses this information to identify regions on the objects surface that require more intense scanning. The scanner could automatically adapt the scanning parameters to the objects material, color, surface and size.

5.2.6 Support for more Open Standards

It would certainly make sense to integrate the Point Cloud Library [RC11] into the next version. This would allows to use different point-cloud to surface-mesh algorithms, besides the powercrust algorithm. Additionally, the scanner could be made OpenNI [Ope11] compatible.

Appendix A

SUS - A quick and dirty usability scale

System Usability Scale

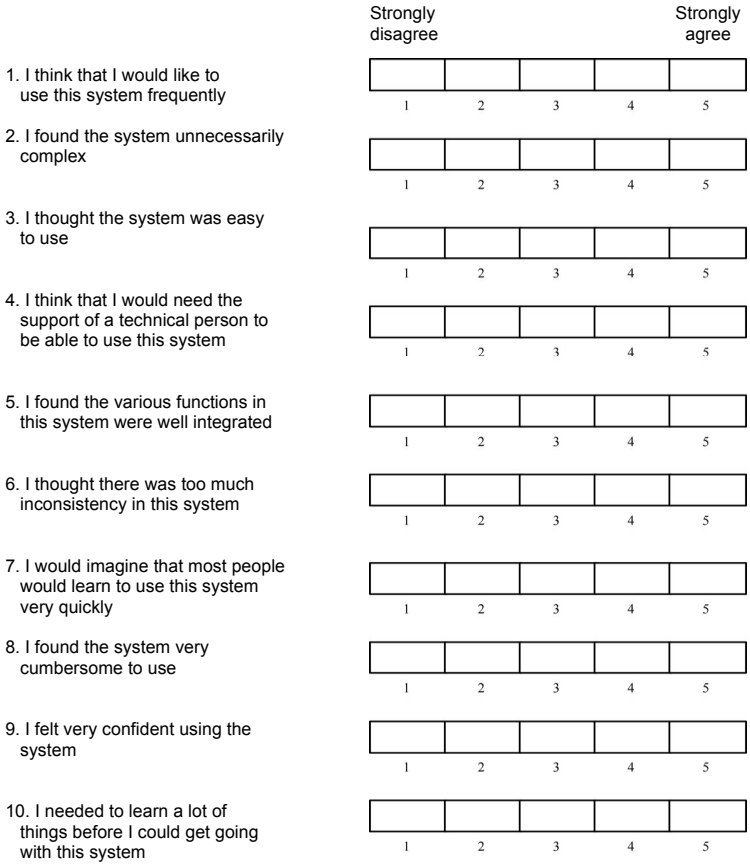

Figure A.1: SUS questionnaire

Appendix B

Schematics for the Laser Cutter Parts

The schematics files for the laser cutter parts were to numerous for publishing in the appendix. The files are provided under the following download link. The files are updated when there is a new version available.

FabScan Laser Cutter Parts [a]

[a] http://hci.rwth-aachen.de/tiki-download_wiki_attachment.php?attId=1388&download=y

Appendix C

Arduino FabScan Shield Schematic and Board Layout

.

The schematics files for the PCB are provided under the following download link. The files are updated when there is a new version available.

PCB Eagle Files [a]

[a] http://hci.rwth-aachen.de/tiki-download_wiki_attachment.php?attId=1390&download=y

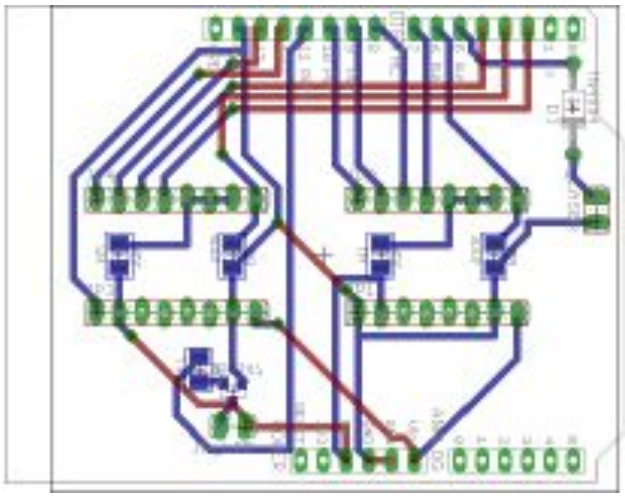

Figure C.1: PCB board

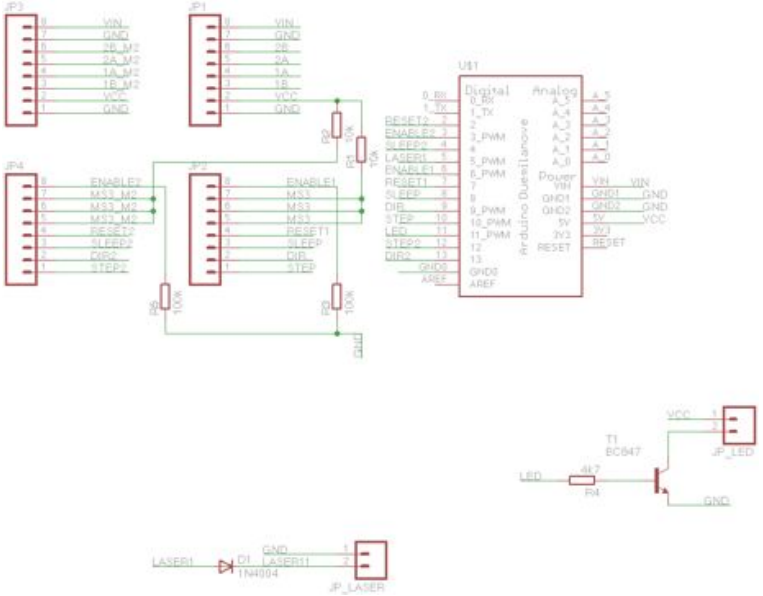

Figure C.2: PCB schematics

Appendix D

Schematics for the 3D Printer Parts

The files for the 3D printer parts are provided under the following download link. The files are updated when there is a new version available.

FabScan 3D Printer Parts [a]

[a] http://hci.rwth-aachen.de/tiki-download_wiki_attachment.php?attId=1389&download=y

Appendix E

Digital Content

The attached DVD-ROM contains the source code and the executables of ”FabScan” as well as the firmware for the Arduino. In addition, the DVD contains the point-clouds and surface-meshes shown in the figures. All the files needed to built a hardware copy of the FabScan are also included.

Bibliography

[A94] LAURENTINI A. The Visual Hull Concept for Silhouette-Based Image Understanding. In *IEEE TPAMI*, pages 150–162, March 1994.

[AC01] N Amenda and S Choi. *The power crust*. The proceeding of the ACM symposium on solid . . . , 2001.

[BK08] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 1st edition, October 2008.

[BKM08] Aaron Bangor, Philip T Kortum, and James T Miller. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction*, 24(6):574–594, July 2008.

[Bla04] F Blais. Review of 20 years range sensor development. *Journal of Electronic Imaging*, September 2004.

[Bor01] Jan Borchers. *A Pattern Approach to Interaction Design*. Wiley, 1 edition, May 2001.

[BP98] J.-Y. BOUGUET and P. PERONA. 3d photography on your desk, September 1998.

[Bro11] John Brooke. SUS - A quick and dirty usability scale. Technical report, September 2011.

[Car92] JM Carroll. Getting around the task-artifact cycle. *ACM Transactions on Information Systems . . .* , 1992.

[CR08] M Corsini and G Ranzuglia. Meshlab: an open-source 3d mesh processing system. *ERCIM News*, 2008.

[FF11] STL File Format. http://mech.fsv.cvut.cz/dr/papers/ Lisbon04/node2.html, October 2011.

[For05] USB Implementers Forum. Universal Serial Bus Device Class Definition for Video Devices, June 2005.

[Ger05] Neil Gershenfeld. *FAB: The Coming Revolution on Your Desktop–From Personal Computers to Personal Fabrication*. Basic Books, April 2005.

[GSGC08] Diego Gutierrez, Veronica Sundstedt, Fermin Gomez, and Alan Chalmers. Modeling light scattering for virtual heritage. *Journal on Computing and Cultural Heritage (JOCCH*, 1(2), October 2008.

[HMK82] E.L. Hall, C.A. MCPherson, and Tio J B K. Measuring curved surfaces for robot vision. *Computer*, 15(12):42–54, September 1982.

[HS97] J Heikkila and O Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, page 1106, February 1997.

[LIAI11] A Legarda, A Izaguirre, N Arana, and A Iturrospe. A new method for Scheimpflug camera calibration. *Electronics, Control, Measurement and Signals (ECMS), 2011 10th International Workshop on*, pages 1–5, 2011.

[LMW+11] T Leyvand, C Meekhof, Yi-Chen Wei, Jian Sun, and Baining Guo. Kinect Identity: Technology and Experience. *Computer*, 44(4):94–96, 2011.

[LT09] Douglas Lanman and Gabriel Taubin. Build your own 3D scanner: 3D photography for beginners. *SIGGRAPH '09: SIGGRAPH 2009 Courses*, August 2009.

[MLGG02] B Mikhak, C Lyon, T Gorton, and N Gershenfeld. Fab Lab: An alternate model of ICT for development. *... on Development by ...*, 2002.

[NN94] K. S. NAYAR and Y NAKAGAWA. Shape from focus. In *IEEE Trans. Pattern Anal. Mach. Intell. 16*, pages 824–831, April 1994.

[Ope11] NI Open. Introducing OpenNI www.openni.org, September 2011.

[PM83] F.J. Pipitone and T.G Marshall. A wide-field scanning triangulation rangefinder for machine vision. *International Journal of Robotics Research*, 2(1):39–49, September 1983.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, 2011.

[Rog11] Rick Rogers. Kinect with Linux. *Linux Journal*, 2011(207), July 2011.

[SPSB04] J. SALVI, J PAGE S, and J. BATLLE. Pattern codification strategies in structured light systems. In *Pattern Recognition*, pages 827–849, September 2004.

[SRDD06] S SEITZ, SZELISKI R, CURLESS B DIEBEL, and J SCHARSTEIN D. A comparison and evaluation of multi-view stereo recon- struction algorithms. In *CVPR 2006*, June 2006.

[Uno11] Arduino Uno. http://arduino.cc/en/, October 2011.

[WLZ10] Y. M Wang, Y Li, and J. B Zheng. A camera calibration technique based on OpenCV. In *Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on*, pages 403–406, 2010.

[WMMW06] Simon Winkelbach, Sven Molkenstruck, and Friedrich M Wahl. Low-Cost Laser Range Scanner and Fasst Surface Registration Approach. *DAGM*, pages 1–11, September 2006.

[WN98] M. WATANABE and S. K. NAYAR. Rational filters for passive depth from defocus. In *Int. J. Comput. Vision 27*, pages 203–225, September 1998.

[XRL11] Yuan Xin, Zhu Ruishuang, and Su Li. A Calibration Method Based on OpenCV. *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, pages 1–4, 2011.

[Zha00] Z Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

Index

Typeset October 5, 2011