RWTH AACHEN UNIVERSITY

FabScan Pi - an open-hardware stand-alone web-enabled 3D scanner

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

by
Mario Lukas

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr.-Ing. Stefan Kowalewski

Registration date: Feb 18th, 2015
Submission date: July 09th, 2015

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 2015
Mario Lukas

Contents

List of Figures

List of Tables

Abstract

3D scanners become increasingly important because they can be used for digitizing physical objects in shape and color. Those digitized objects can be 3D printed. In 2011 the FabScan 3D scanner started out as a bachelor thesis [Eng11]. The FabScan 3D scanner is characterized by it's usability and open source hardware and software. Even though it's open source software there was no progress by developing new software features in the FabScan community. The FabScan software does not run on modern operating systems anymore.

The aim of this thesis is to find a solution for this problem. The result - the FabScab Pi - is a stand-alone web-enabled 3D scanner which is based on the hardware parts of the FabScan CUBE. The electronic components are replaced by a Raspberry Pi 2 with the Raspberry Pi camera module and a FabScan Pi HAT.

A new software was developed which consists of two parts. The first part is the back-end server and the second part is the web-enabled user interface. To keep the software development simple widely-used script languages like Python and JavaScript are used.

It was possible to solve the most mentioned shortcomings of the old FabScan software by using different web technologies and existing software like MeshLab, WebGL and Motion-JPEG.Among other things improvements of the hardware, like adding LED's for a better ambient lighting, leaded to better scan results.

Überblick

3D Scanner gewinnen immer mehr an Bedeutung, da mit ihnen digitale 3D Modelle von physikalischen Objekten erzeugt werden können, die sich für beispielsweise 3D-Druck verwenden lassen. Im Jahr 2011 wurde im Rahmen einer Bachelorarbeit der FabScan 3D Scanner entwickelt [Eng11]. Dieser kostengünstige Bausatz zeichnet sich durch seine einfache Bedienbarkeit und die Offenlegung der Baupläne aus. Obwohl die Software open source ist, gab es in der FabScan Community in den letzten Jahren keine Beiträge, die das Programm mit den nötigen Updates versorgt hätten. Auf modernen Betriebssystemen kann man die FabScan Software nicht mehr verwenden.

Ziel dieser Arbeit ist es, eine Lösung für dieses Problem anzubieten. Das Resultat - der FabScan Pi - ist ein stand-alone web basierter 3D Scanner, basierend auf dem Gehäuse des weitverbreiteten FabScan CUBE. Die elektronischen Komponenten wurden durch einen Raspberry Pi mit Raspberry Pi Kamera Modul und einem eigens entwickelten FabScan Pi HAT ersetzt.

Die Software wurde neu entwickelt und besteht nun aus zwei Teilen, dem Backend Server und der web basierten grafischen Benutzerschnittstelle. Um die Weiterentwicklung der Softwarekomponenten zu vereinfachen, wurden die Spachen Python und JavaScript eingesetzt, die einen einfachen Einstieg ermöglichen und in der Community stark verbreitet sind.

Durch den Einsatz unterschiedlicher Technologien und Software, wie MeshLab, WebGL und Motion-JPEG, konnten einige der genannten Mängel der alten FabScan Software behoben werden. Unter anderem führten Änderungen an der Hardware, wie das Einführen von kontrollierten Lichtverhältnissen, zu höherwertigen Scanergebnissen.

Acknowledgements

I would like to thank Prof. Dr. Jan Borchers and Prof. Dr.-Ing. Stefan Kowalewski for giving me the possibility to do this work. Special thanks go to my supervisor Renè Bohne and to all the people of the FabLab Aachen for their constant support and useful input. Special thanks also to Christoph Emonds for some code reviews during my work on this thesis, Volker Bombien for checking the language and spelling of this thesis and Stephan Watterott for supporting me with hardware. Finally I want to thank my girlfriend Ruth Stiefelhagen for being patient with me the last month while I have besieged our workroom.

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 American English.

Download links are set off in coloured boxes.

File: myFile[a]

[a] https://github.com/username/projectname/file_number.file

Chapter 1

Introduction

1.1 What is 3D Scanning and What is it Good for?

A 3D Scanner is a hardware and software system for digitizing the shape and color of physical objects under known environmental conditions. The digitized data can then be used to construct digital three-dimensional models.

The usage of digitized data or three-dimensional models created by 3D scanners comprises industrial design, medical diagnosis, cultural heritage, multimedia, entertainment, rapid prototyping, and digital fabrication.

Different technologies and approaches are available to perform 3D scanning, including structured light, coded light, time of flight and more [Bla04]. The most common technologies and approaches are already described by detail in the Thesis of Francis Engelmann [Eng11]. Each technology comes with limitations, advantages and costs.

Some limitations are depending on the kind of objects that should be digitized. For example, line laser based 3D scanners have some difficulties with shiny, mirroring or transparent objects. The most important issue is caused by influence of ambient lighting.

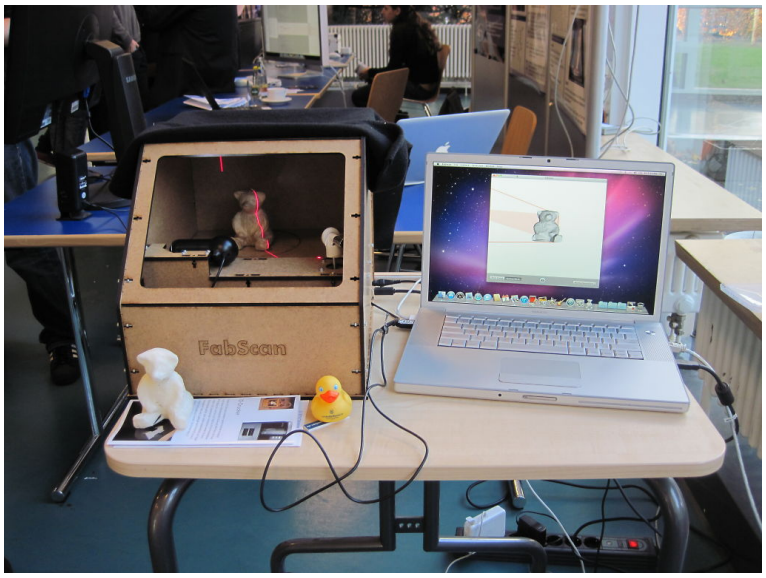

Figure 1.1: The first version of FabScan was the FabScan 100 in 2011.

Since the work of this thesis is based on the FabScan this document will focus on 3D laser scanners.

1.2 FabScan

The FabScan is an open-source, do-it-yourself 3D laser scanner that started out as a thesis project by Fancis Engelmann [Eng11]. The idea behind the FabScan was to supply the FabLab Aachen with a low-cost 3D scanner.

In 2014, development of the project was taken over by Renè Bohne and me. FabScan was featured on Thingiverse[1] and it is supported by a steadily growing user group on Google Groups.

[1] http://www.thingiverse.com

1.3 Motivation

The current FabScan software highly depends on several third party libraries like OpenCV and QT . All those dependencies lead to a complicated release process for different operating systems. Developers and users need advanced programming skills to work on the existing FabScan software. The usage of different operating systems aggravates the software support. The aim of this thesis is to replace the current FabScan software by a new software which runs on a pocket sized mini computer. The new software should be more user and developer friendly. Finally, it should solve the shortcomings of the previous FabScan software and hardware which are specified in chapter 2.5.

1.3.1 FabScan Pi a Web-Enabled Stand-Alone 3D Scanner

FabScan Pi is the next generation of FabScan. The hardware setup is based on the FabScan CUBE . The webcam, Arduino , and FabScan shield are replaced by a Raspberry Pi with the Raspberry Pi camera module and a new developed FabScan HAT for Raspberry Pi. A HAT (Hardware Attached on Top) is a PCB that extends the Raspberry Pi just like an Arduino shield extends the Arduino.

The old FabScan software is replaced by a new server-based software which consists of two parts. A web-enabled user interface and a back-end FabScan API server. Since the new FabScan Pi software is web-enabled the user is able to perform a scan without installing any software parts on a laptop, mobile phone or tablet. The whole scan process can be controlled over a web-enabled user interface. The user is able to perform a scan by calling the FabScan Pi server URL in a web browser.

1.4 Overview

This section gives an overview of the chapters in this thesis and their contents.

- Chapter 2 "Related Work" is split into a software and a hardware related section.

  The first part of the software section introduces some common used web-technologies and tools introduced by the HTML5 web-standard. The second part describes 3D scanning related software and file formats.

  The hardware section gives an overview of existing, affordable 3D scanners. The end of the section will show a tabular overview of the mentioned 3D scanning devices.

- The chapter 3 "Own Work" consists of three parts. The first part starts by listing the requirements for the FabScan Pi project. In the second section the hardware prototypes are presented. Each prototype is accompanied by a short discussion. The third section of this chapter describes the software front- and backend.

- In chapter 4 "Evaluation" we will discuss if the requirements from 3.1 "Requirements" are met.

- Chapter 5 "Summary and future work" contains a summary of this thesis and some ideas for future work.

Chapter 2

Related Work

This chapter consists of two parts. First it starts by giving an overview of some common available desktop 3D scanning devices. In the second part, some HTML5 web-technologies are introduced. Finally the focus is put on MeshLab and the most related 3D scanning file formats.

2.1 Web Technologies

In this section first some web technologies are introduced, which are included in the HTML5 standard. Then a short overview to AngularJS is given.

2.1.1 WebGL

WebGL (Web Graphics Library) is a JavaScript API for web-browsers, which supports hardware accelerated 3D graphics without the usage of plug-ins. WebGL is based on OpenGL ES 2.0 and started out of Canvas 3D, developed by the Mozilla Foundation in 2006. [Vuk07]

WebGL works with any platform that supports OpenGL. WebGL is supported in all major browsers including Internet Explorer from version 11 and it works on various mo-

bile platforms including iOS from version 8. The official WebGL website offers a simple test page[1].

2.1.2 WebSockets

The WebSocket Protocol is provided by the HTML5 standard and defined by rfc6455 [IF11]. It is the next generation of asynchronous communication from client to server. Server and client communicate over a single TCP connection. "The protocol is full-duplex and its header is much smaller than the Header of HTTP. HTML5 WebSockets provide an enormous reduction in unnecessary network traffic and latency compared to the unscalable polling and long-polling solutions that were used to simulate a full-duplex connection by maintaining two connections." [Kaa13] Another effect is that data and notifications can come and go between browsers and Web servers with no delay and no need to arrange for additional requests.

A WebSocket interaction begins with a handshake in which the two parties (browser and server) mutually confirm their intention to communicate over a persistent connection. Next, a bunch of message packets are sent over TCP in both directions. Figure 2.1 outlines how the WebSocket Protocol works.

2.1.3 Motion-JPEG

JPEG stands for the Joint Photographic Experts Group standard, a standard for storing and compressing digital images. Motion-JPEG extends this standard by supporting videos. In motion-JPEG, each frame in the video is stored in the JPEG format.

Motion-JPEG is mostly used by video-capture devices such as digital cameras, IP cameras and webcams. For network cameras the MJPEG codec seems to be ideal, because it needs hardly any hardware resources for encoding the

[1] https://get.webgl.org

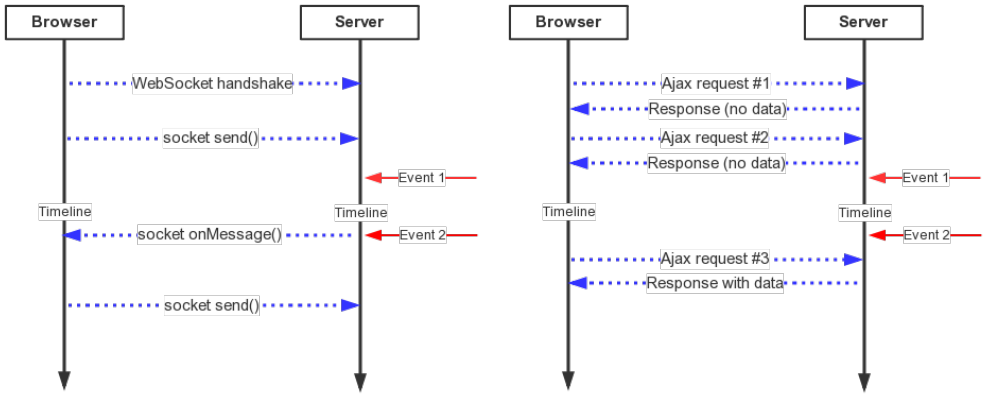

Figure 2.1: WebSocket communication compared to HTTP AJAX polling. Left: WebSockets communication. Right: HTTP with AJAX polling

video and nearly every browser can play MJPEG streams [Inc06].

HTTP video streaming with Motion-JPEG is realized by sending separate images to individual HTTP replies on a specified marker. The client is informed to expect a new frame by a special mime-type. The TCP connection is kept open as long as the client wants to receive new frames and the server wants to provide new frames.

2.1.4 AngularJS

AngularJS[2] is an open source web application framework which is powered by the Google community. AngularJS helps to create single page applications and web applications that don't require anything more than HTML5, JavaScript and CSS. It was created to support web programs with MVC capability which enables easier development and testing.

[2] https://angularjs.org

2.2 3D Scanning related Software

This section introduces software which is used for editing and post processing 3D scanned data. At the end of this section the most common 3D scanning related file formats are described.

2.2.1 MeshLab

MeshLab[3] is an open source, platform independent and extensible software for processing and editing 3D triangular meshes and point clouds. It is mostly used for post processing 3D scans. MeshLab can import and export most of the common 3D file formats. MeshLab is based the VCG library which was developed at the Visual Computing Lab of ISTI - CNR. MeshLab was developed at the Computer Science Department of the University of Pisa during a course. MeshLab provides a set of tools for editing, cleaning, healing, inspecting, rendering and converting point clouds and meshes. The user can create filter chains which can be used to create batch operations on 3D files. Filter chains can be easily exported and imported. MeshLab also provides a command line client called MeshLab Server.

2.2.2 3D File Formats

PLY

PLY is known as Polygon File Format or the Stanford Triangle Format. It was designed to store three-dimensional data produced by 3D scanners.

A PLY file consists of a header followed by a list of vertices and a list of polygons. The number of vertices and polygons in the file are defined in the PLY header. The header also states what properties are associated with each vertex such as coordinates (x, y, z) , normals and colors.

[3] http://meshlab.sourceforge.net

The file format has two sub-formats: an ASCII representation for easily getting started, and a binary version for compact storage and for rapid saving and loading [Bou15].

STL

The Stereo Lithography file format (STL) was introduced by 3D Systems and is also known as Standard Tessellation Language. STL files are widely used in Computer Aided Manufacturing (CAM) and rapid prototyping.

STL is very useful for 3D printing because it describes only the surface geometry of a three-dimensional object without any representation of color, texture or other common CAD model attributes. STL files can be represented in ASCII or binary format. "The surface is tessellated or broken down logically into a series of small triangles (facets). Each facet is described by a perpendicular direction and three points representing the vertices (corners) of the triangle." [Bur89]

X3D

X3D stands for "Extensible 3D" and is a XML-based file format, defined by Web3D Consortium and the World Wide Web Consortium (W3C). "It is an ISO ratified standard that provides a system for the storage, retrieval and playback of real time graphics content embedded in applications, all within an open architecture to support a wide array of domains and user scenarios." [Web15]. It is the third-generation successor to the Virtual Reality Modeling Language (VRML). Nowadays the most widespread web browser support the X3D format by default.

X3D files can be exported by MeshLab and it supports colored meshes. With the X3D file format 3D print services are able to provide printing of colored objects.

2.3 Existing 3D Scanners

In this different 3D Scanners are presented, which are available on the market now. As the FabScan is a line laser scanner in the following section the focus is set to similar devices.

To perform a scan the object has to be placed on the turntable. Then the laser is switched on and off while the webcam is taking pictures of the object. With a mathematical process, called triangulation [LT09], the depth data can be calculated out of the taken pictures. To perform a full scan the turntable has to do a rotation of 360 degrees. The process continues for each turntable position. If the scan process is finished the object is digitized and available as a point cloud.

At the end of this section all devices are compared in a tabular.

2.3.1 FabScan CUBE

FabScan CUBE is a later version of the previous FabScan which is described in the FabScan thesis. FabScan consists of a line laser, a webcam, a turntable and an Arduino based controller board for scanning. The FabScan works like many other 3D laser scanners, but is aided by the incorporation of an enclosure that helps to even out light levels, preventing distortion in the scan. FabScan CUBE "Do-It-Yourself 3D Laser Scanner" Kits are available for 130 Euro.

2.3.2 David Scanner

David Scanner[4] is a collection of software and hardware components. Different 3D scan approaches are provided by David. The David Starter Kit in version 2 is available for

[4] http://www.david-3d.com/en/products/starter-kit-2

Figure 2.2: FabScan CUBE

585 Euro. The David Scanner Software which is contained in the Starter Kit is proprietary and runs only on Windows. Webcam and a line laser module are delivered within the Starter Kit. By using the DAVID 3 software the user can manually merge multiple point clouds from different scans and convert them to a surface mesh by hand.

2.3.3 MakerBot Digitizer

MakerBot Digitizer[5] is a 3D Scanner introduced by MakerBot Industries in 2013. The Digitizer scans objects with two line lasers and a camera. The object has to be placed on a rotary platform in front of the camera. Objects can have a height and diameter of 20cm. The Digitizer is available for 950 Euro. A full scan in medium resolution is done by stepping the rotary table 800 times in 12 minutes. After the

[5] https://store.makerbot.com/digitizer

Figure 2.3: David Scanner

Figure 2.4: Makerbot Digitizer

scan the digitizer is able to mesh the point-cloud to a STL file. MakerBots Digitizer is not able to scan the color or texture of an object. The scan software, called Makerware for Digitizer, is proprietary and runs only on Windows.

Figure 2.5: Matter Form 3D Scanner

2.3.4 Matter and Form 3D Scanner

The Matter and Form 3D Scanner[6] works like most of the other line laser scanners. It offers two line lasers and a build-in camera module. The camera and laser module can be moved up and down, for scanning larger objects. Objects have to be placed on the rotating platform for scanning. The Matter and Form 3D Scanner can fold up it's base when it is not used. It is delivered fully assembled and available for $599.99. The Matter and Form 3D Scanner can perform a scan in 5 minutes and it provides color scans. Higher resolution scans may require several hours. The software of Mater and Forms 3D Scanner runs under Windows and Mac OS.

[6] https://matterandform.net/scanner

Figure 2.6: BQ Cyclop

2.3.5 BQ Cyclop

BQ's 3D Scanner Cyclop[7] was introduced at CES in 2015. The Cyclop uses two red line lasers and a standard USB webcam. The Cyclop is controlled by a custom Arduino based controller. BQ has developed their own desktop 3D scanning application called Horus[8]. Horus is open source and written in Python and runs on Windows and Linux based operating systems.

2.3.6 Atlas 3D Scanner

The Atlas 3D Scanner[9] was funded by Kickstarter. The Atlas 3D Scanner uses two red line lasers modules and a webcam to scan an object on an rotating platform. The Atlas 3D Scanner replaces the Arduino with a Raspberry Pi. The Atlas 3D Scanner is open-source and it's software is called FreeLSS. FreeLSS is written in C++ and runs under Linux on a Raspberry Pi. FreeLSS is web-enabled and can be controlled with a browser. FreeLSS can generate results as PLY, XYZ and STL. The web application consists of three dialogs, Main Interface, Camera Feed from Scanner and

[7] http://www.bq.com/gb/ciclop
[8] https://github.com/bq/horus
[9] http://www.freelss.org

Figure 2.7: Atlas 3D Scanner

Settings Page. It does not provide a real-time preview of the scanned point cloud.

2.4 Comparing Existing 3D Scanners

	Do-It-Yourself	Ready-To-Print	Web-enabled	Portability	Usabillity	Standalone Device
FabScan CUBE	✓	✗	✗	✗	✓	✗
David Scanner	✗	✓	✗	✗	✗	✗
Makerbot Digitizer	✗	✓	✗	✗	✓	✗
Matter and Form	✗	✓	✗	✗	✗	✗
BQ Cyclop	✓	✗	✗	✓	✓	✗
Atlas 3D Scanner	✓	✗	✓	✗	✗	✓

Table 2.1: Scanner comparison

2.5 Shortcomings of the FabScan CUBE

The FabScan CUBE should be used as base system for this thesis. Since the first FabScan was released in 2011, some hardware and software issues came up over the years. The following list shows the shortcomings of the current FabScan. These aspects were discussed by users of the FabScan user group at Google and on some FabScan community blogs.

2.5.1 Scan Speed

A scan in medium resolution takes about 15 minutes. Improving the scan speed is one of the suggested features in the FabScan user group. As the scanning process is running in a single thread, more performance should be possible by splitting it up to multiple processes.

2.5.2 Post Processing of Scans

The current version of the FabSacn software is not able to export clean mesh files for 3D printing. Most users use MeshLab and it's filters for cleaning up point clouds and export it to the needed mesh format.

2.5.3 Scan Quality

Scanning the object texture highly depends on environment lighting. The current FabScan software does not provide the possibility to control the light during the scan process. Also the laser line detection highly depends on the environmental light. By using the current FabScan software and changing the environment light it happens that the best light for scanning the texture excludes the best light for laser line detection.

The current software has some major bugs which lead to a software crash by choosing the different scan quality levels.

2.5.4 Usability

Settings can be set in a separated settings dialog. The user has to test those settings by performing a new scan. While each scan takes 15 minutes in medium quality, changing the settings takes time and effort. The settings are highly depending from the objects color, surface and environmental light.

2.5.5 Build Process and System Dependencies

Developers of the FabScan community mentioned that the tool chain to build the FabScan software is too complicated and depending on special versions of third party software. It is hard to compile the software on different operating systems. It is not possible to get the current software working with all its features under Windows.

2.5.6 Summary of Related Work

In this chapter web technologies, existing 3D scanners and the mentioned shortcomings of the FabScan are introduced. These results are used to derive the requirements for the new FabScan hard- and software in the next chapter.

Chapter 3

Own Work

This chapter is introduced by deriving the system requirements out of the shortcomings mentioned in section 2.5 “Shortcomings of the FabScan CUBE”. In the second part of this chapter the new hardware prototypes will be presented and discussed. In the last section the software design and architecture will be described.

3.1 Requirements

The FabScan Pi should remain an affordable 3D scanner for hobbyists and maker. Another aim is to improve the hardware and software to get the best possible scan quality. The comparison of the existing 3D scanners in 2.4 “Comparing Existing 3D Scanners” and the shortcomings of the FabScan in 2.5 “Shortcomings of the FabScan CUBE” result in the following requirements:

- **Affordability of FabScan Pi** The budget of the hardware for the new FabScan should be nearby as affordable as the hardware that is used for FabScan CUBE.
- **Do-it-Yourself** The new FabScan should stay a device which anyone can build with access to a FabLab or by ordering an inexpensive FabScan construction kit.

- **Usability** Untrained users should be able to cope with the new system. The user should be able to perform a high quality scan with a minimum number of clicks, time and effort. Users of the old FabScan should be able to use the new FabScan software without much prior knowledge.

- **Provide Settings Preview** The user should be able to change the settings for the best scan quality assisted by the software.

- **Save and load scan settings** Object specific scan settings should be automatically saved. If the same object or an object with similar properties is scanned again, the user should be able to load settings of a previous scan.

- **Stand-alone Device** To provide a device with the same hardware and software setup for all users, a small Linux based embedded computer system should be used to run the FabScan software. The new FabScan should be usable as a stand-alone hardware device.

- **Web-enabled User Interface** With the requirement of a stand-alone device the system should be controllable without the usage of additional built-in hardware. Therefore, the system should provide a web-enabled user interface which can be used on all devices with a web-browser.

- **Downloading Scans in Different Formats** The user shall be able to download scans in different file formats.

- **Integration of MeshLab** The software should be able to clean up point-clouds and export watertight meshes in a printable format. MeshLab should be integrated in the new software to provide a configurable point-cloud post processing mechanism.

- **Improving scan quality** The scan quality of FabScan should be improved by new hardware and software features.

- **Improving scan speed** The scanning speed should be increased by introducing multi processing scan algorithms.

3.2 Hardware

In this section several DIA-cycle (Design Implement Analyze) [Car92] [Bor01] iterations will lead to a final hardware prototype . Every prototype iteration will be tested with a set of objects with different color, shape, surface and material properties. After each iteration it is discussed what was learned and how to improve the results with the next prototype iteration. The software part was developed during the hardware iterations.

3.2.1 First Hardware Prototype

The first iteration is based on the FabScan CUBE, presented in 2.3.1 "FabScan CUBE". For this prototype a Raspberry Pi 2 was added to the existing FabScan CUBE setup. The Raspberry Pi2 has a quad core processor and should provide enough power for image processing. Therefore the Logitech C270 webcam is replaced by a Raspberry Pi camera module. With the usage of the Raspberry Pi camera module there is no need to disassemble a webcam.

Definition:
Raspberry Pi

RASPBERRY PI:
"Raspberry Pi is a small, single-board computer developed for computer science education. A United Kingdom (UK) charitable organization called the Raspberry Pi Foundation developed the device.
Raspberry PI is about the size of a credit card, has a 32-bit ARM processor and uses a Fedora distribution of Linux for its default operating system (OS). It can be programmed with Python or any other language that will compile for ARM v6." [Rou12]

The Arduino with the FabScan shield on top is connected

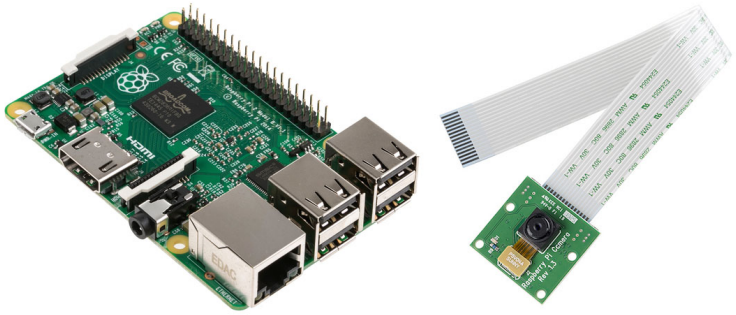

Figure 3.1: Left: Raspberry Pi 2. Right: Raspberry Pi camera module

to the USB port of the Raspberry Pi 2. An additional 5V power supply is added for powering the Raspberry Pi 2. The first software prototype was written in Python. For this first Python based server prototype a rough web-enabled user interface was designed for testing basic scanning operations on a Raspberry Pi 2.

Discussion

A scan with the first prototype showed that the Raspberry Pi 2 in combination with the FabScan CUBE components, is suitable for fast image processing and 3D scanning. A quality improvement to the FabScan CUBE was given by using the Raspberry Pi 2 camera module. The camera module has a maximum resolution of 2592 x 1944 pixels [Jon14]. The Logitech c270 webcam, used by the old FabScan CUBE setup provided a maximal resolution of 1280 x 720 pixels. Therefore the first quality improvement is done by doubling the camera resolution what results in a higher point cloud density. Also the assembly of the hardware setup is easier for the user, because the camera module is ready to

use with the Raspberry Pi 2. The camera consists of a small (25x20x9mm) circuit board, which connects to the Raspberry Pi's Camera Serial Interface (CSI) bus connector via a flexible ribbon cable. After it is connected to the Raspberry Pi, it has to be screwed to the FabScan case front. The wiring of this prototype is overhead. Also the USB connection to an additional Arduino board with a FabScan shield can be removed. Because the object shifts when the turn table is rotating, the object has to be glued to the table with some tape.

3.2.2 Second Hardware Prototype

For the second prototype the Arduino and the FabScan shield are replaced by a Raspberry Pi FabScan HAT, shown in Figure 3.2.

The FabScan HAT carries the motor drivers (1) for the turntable and laser stepper motor. The FabScan HAT can carry up to four motor drivers. The motors can be easily connected to screw terminals, which are positioned in front of the motor drivers (3). A second laser and an LED port was added to the FabScan HAT (2). The FabScan HAT is powered by a 12 Volt power supply, which can be connected to a power jack (5). A small step-down circuit placed on the HAT ensures that the Raspberry PI 2 is powered with 5 Volt. An ATMega328 IC with Arduino Firmware is placed on the HAT (4). The FabScan firmware is flashed to the ATMega328. The Arduino FabScan firmware communicates with the Raspberry Pi 2 over the GPIO serial connection. The Pololu motor drivers are replaced by silent step stick motor drivers for smoothing the motor rotations to prevent object shifting while the table is rotating. Some white LEDs are placed beside the camera to get a better texture scan.

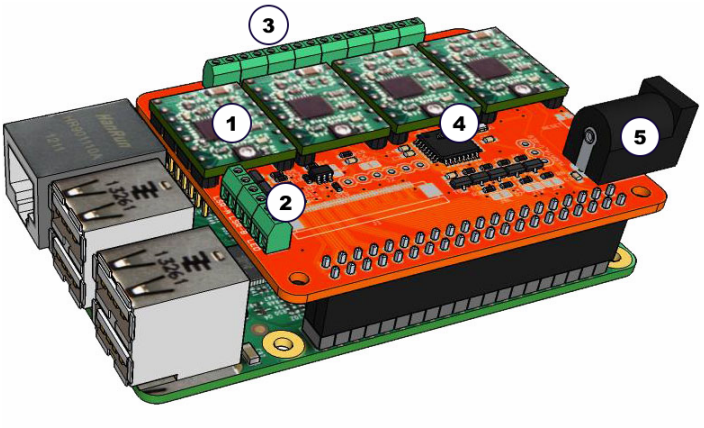

Figure 3.2: FabScan Pi HAT mounted on Raspberry Pi 2

Definition:
Raspberry Pi HAT

RASPBERRY PI HAT:
"A HAT is an add-on board for Raspberry Pi B+ that conforms to a specific set of rules that will make life easier for users. A significant feature of HATs is the inclusion of a system that allows the B+ to identify a connected HAT and automatically configure the GPIOs and drivers for the board, making life for the end user much easier." [Ada14]

To solve the ambient lighting issue, different colored transparent foils were placed in front of the camera. The idea was to get a higher laser contrast what should improve the laser line detection a lot.

Discussion

With this second prototype the wiring was cleaned up and the assembly process is much more easy. No additional Arduino or FabScan Shield are connected to the USB port of the Raspberry Pi 2. Just the Raspberry Pi 2 HAT has to be mounted to the GPIO port of the Raspberry Pi.

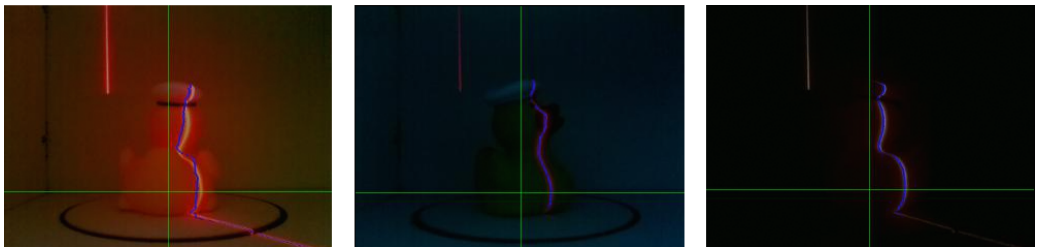

Figure 3.3: Different colored filter foils in front of the camera module. Left: without filter. Middle: with blue filter. Right: with red filter.

Figure 3.3 shows the results of using different filters for the laser line detection. On the left picture no filter foil was used. The laser line detection is poor. The recognized blue line lays beside the real laser line. The detected line also consists of some scattering. For the picture in the middle a blue filter foil was used. The laser line detection is much better than in the left image. But also the blue foil doesn't serve the best recognized laser line detection. On the right picture a red filter foil is used. The red foil leads to the best laser line detection result. Therefore it was shown that the laser line recognition can be improved by changing the image contrast by using different filter foils.

Another issue was solved by adding white LEDs beside the Raspberry Pi camera module. With those white LEDs the colors and texture of a scanned object is much clearer and brighter. Further every object texture can be scanned with the same ambient light conditions.

But a new issue came up by using the filter foils. The scan process has to be interrupted for manually removing the foil to get a clear camera image for scanning the texture and color of the object.

3.2.3 Final Hardware Prototype

For the third Hardware Prototype a RGB LED ring with 12 LEDs was added around the Raspberry Pi 2 Camera module. The camera module was also screwed to a spring mounted plate. By turning the screws of the spring

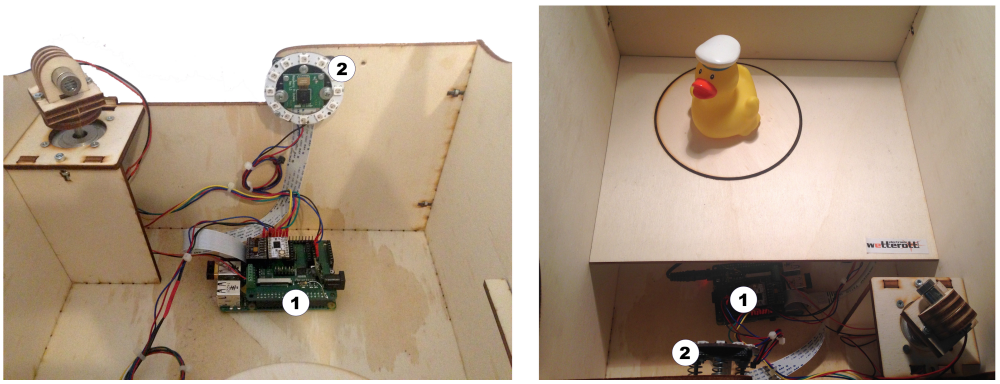

Figure 3.4: The final hardware prototype. Left: view from back to front. Right: view from front to back. (1) Raspberry Pi with FabScan Pi HAT. (2) Raspberry Pi camera module with mounted LED ring.

mounted plate it is easier for the user to calibrate the camera module. The RGB led replaced the colored foils. By illuminating the object with different RGB colors nearly the same contrast effect is reached as by using different colored foils. Furthermore the white colored LEDs used for the second hardware prototype can be removed, because the white color can be matched by turning all RGB values to the maximum.

Discussion

The third and final hardware prototype solves most of the hardware issues. The RGB LEDs helped to control the color contrast level and to improve the laser line recognition like the colored foils of the second hardware prototype did. The efforts led to the fact that the user is not constrained to manually remove the colored foil before each color and texture scan.

With using the RGB LED ring instead of the white LEDs and the color foils, a scan with texture and colors occurs much better. Figure 3.5 shows two colored point clouds compared to each other. The left point cloud shows a scan without LED lighting. The right point cloud shows a scan where the RGB values of the LED ring are switched to white

Figure 3.5: Scan with different LED light settings. Left: without LED light. Right: with white LED light

light during the scan.

By using the RGB LED ring the laser line recognition can be improved by setting the best contrast light value for the object which should be scanned.

3.3 Software

In this section the software parts of FabScan Pi are described. The first part of this section describes the general architecture with all its layers. The second part explains the different communication protocols of FabScan Pi. In the third part of this chapter the back-end architecture and its modules are described. Finally, in the last part the focus is set on the web-enabled user interface.

3.3.1 General Architecture

Figure 3.6 shows the different layers of the FabScan Pi software architecture. The base of the software is represented by the Arduino compatible firmware of the FabScan Pi HAT which is described in 3.2.2 “Second Hardware Prototype”. The second layer consists of the back-end server.

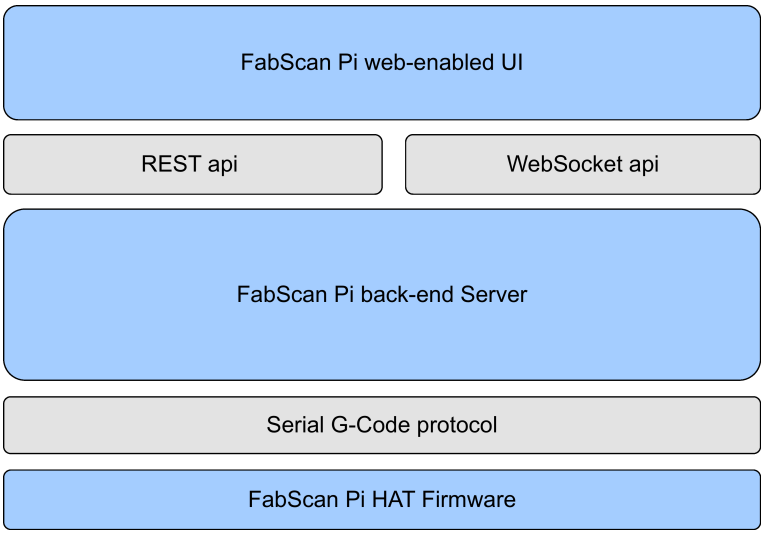

Figure 3.6: FabScan Pi software architecture

The communication between the firmware and the back-end server is done by a G-Code based protocol described in 3.3.2 "Hardware Control Protocol". The top layer of the FabScan Pi software is formed by the user interface. Back-end and user interface communicate over a WebSocket and a REST API. The protocols are described in 3.3.2 "Protocol Definitions"

3.3.2 Protocol Definitions

The FabScan Pi Software communicates over three different protocols. One protocol is for controlling the FabScan hardware over the back-end server. It is implemented in the FabScan Pi HAT's firmware. The other protocols are used for the higher level communication between the back-end server and the web-enabled user interface.

Hardware Control Protocol

The protocol to control the hardware is implemented in the FabScan Pi HAT firmware. When the FabScan Pi server is

started for the first time it checks by a ping if the firmware is already flashed to the FabScan Pi HAT. If it's not flashed the back-end server starts the firmware flashing process by calling a Python firmware flash routine. This process guarantees that the Raspberry Pi HAT is flashed with the current firmware. It is no longer required that the user has to flash the firmware by himself with the Arduino IDE.

The firmware protocol is kept as simple as possible. Since G-Code , also known as RS-274, [Mes00] is an easy to use and human readable programming language, a custom G-code like protocol is chosen for the FabScan Pi software. With the FabScan Pi G-Code it is easy for the user to debug the hardware functionality by sending the G-Code commands in plain text by using a serial connection. Table 3.1 gives a short functional protocol overview.

A G-Code protocol used for controlling the FabScan hardware.

Command	Argument	Description
G01	T$< n >$ L$< n >$	Turn table or laser for $< n >$ steps
G06	-	Start turning table
G07	-	Stop turning table
G100	-	Help
M05	R$< n >$ G$< n >$ B$< n >$	Set Led RGB value
M21	-	Laser on
M22	-	Laser off

Table 3.1: FabScan Pi G-Code

Back-end Protocols

The back-end server communicates by using two different protocols with the user interface. The first protocol is a simple WebSocket protocol, it is used for sending and receiving control commands and for transmitting the point cloud fragments during the scan. The WebSocket connection is opened when a client connects to the back-end server. It is hold open until a client disconnects from the server. WebSockets give the opportunity to synchronize data between all connected clients. This mechanism is used to send the same point cloud synchronous to all connected clients while a scan is performed.

A JSON based WebSocket message protocol is used for representing live scan data.

The FabScan Pi WebSocket protocol is split into command and data messages. Those messages are encoded as JSON (JavaScript Object Notation) strings. Command messages in general are containing scanner control commands. Data messages contain scan data. The following section describes the WebSocket message protocol.

Structure command Messages:

```
{
    "type": "COMMAND"
    "data": {
        "command": "<scanner_command>"
    }
}
```

type	The command message type is always defined by the word $COMMAND$
data	Data is an object with different attributes. A minimal data object contains the *command* attribute.
command	The command attribute contains a command for the FabScan back-end. Possible commands are $SCAN, START, STOP$ and $UPDATE\_SETTINGS$
<additional>	The data object can contain additional custom attributes. If command $UPDATE\_SETTINGS$ is used, data should contain a attribute $settings$ which contains the scan settings.

Table 3.2: Description of command message elements

Structure data messages:

```
{
    "type": "ON_NEW_PROGRESS"
    "data": {
        "points": "<points>",
        "resolution": "<value",
        "progress": "<value"
    }
}
```

type	Type defines which type of data the message contains. Possible values are $ON\_NEW\_PROGRESS$ and $INFO\_MESSAGE$
data	The data object contains the message data. The message data can be found in different data attributes. Possible attributes are $points$, $progress$, $resolution$ and $message$
point	An array with point cloud points of the current scan position
progress	Current scan progress means number of current picture
resolution	Total number of pictures for a full scan
message	Message is mostly used for type $INFO\_MESSAGE$ and contains a system message. Example gives $SCAN\_COMPLETE$, $SCAN\_STOPPED$, $SCAN\_STARTED$ etc.

Table 3.3: Description of data message elements

A REST API is used for handling big data events.

The second protocol is implemented as a RESTful (Representational State Transfer) API. The REST API is used for loading, downloading and uploading big data. The user interface calls an URL to perform a REST API operation. The back-end server interprets the contained data and sends a JSON response string to the user interface. Until now, the FabScan Pi REST API contains only a basic set of operations for handling files and folders of scans.

The following GET and DELETE requests operate on the data of a scan. With GET the web-enabled user interface can load all scan related data for a given scan id. With DELETE all related scan data is removed from the FabScan Pi server.

```
GET:
Request:
http://fabscan.local/api/v1/scans/<scan_id>

Response:
{
    "id" : <id>
```

```
    "date" : <date>
    "point_cloud": <ply_file_url>
    "preview_thumbnail": <jped_file_url>
    "scan_settings": <settings_file_url>
    "meshes": {
        "stl": <stl_file_url>,
        "x3d": <x3d_file_url>
    }
}

DELETE:
Request:
http://fabscan.local/api/v1/scans/<scan_id>

Response:
{
    "messgae": "Object removed"
}
```

The following Request can be used to get a full list of scans.

```
GET:
Request:
http://fabscan.local/api/v1/scans

Response:
{
    [
        {
            "id": <scan_id>
            "href": http://fabscan.local/
               api/v1/scans/<scan_id>
        }
    ]
}
```

3.3.3 Back-end server

This section describes the back-end server architecture. Figure 3.7 gives an overview of the main components and modules of the FabScan Pi back-end server.

Python as the language for the back-end server development.

The back-end server is written in Python. Python is a widespread and interpreted, object-oriented, high-level programming language with dynamic semantics. "Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception" [pyt15].

Since one of the ideas behind the Raspberry Pi was to enable people to explore computing and learn to program in languages like Python, this language is a good choice for the FabScan Pi back-end. With the choice of Python also less skilled programmers are able to help developing on the FabScan Pi software. Furthermore there are many libraries available for Python, such as OpenCV, which is used for the image processing parts of the FabScan Pi software. Libraries can be easily installed by using Python's packet manager pip. Python runs on all common operating systems.

Figure 3.7 shows the different modules of the FabScan Pi back-end server. The scan processor is the main server thread of the FabScan Pi back-end software. It contains a state machine and the hardware controller. The state machine is responsible to interpret the received commands from the front-end. All commands are received by the WebSocket module and brought to the state machine by the event bus. Then the received command is parsed and executed by the state machine. The related protocol is defined in 3.3.2 "Back-end Protocols".

The hardware controller encapsulates the hardware connection and its protocol. It acts like a wrapper between hardware and state machine and creates a serial connection to the FabScan Pi firmware. Over this connection the scan processor is able to send commands to the hardware as described in 3.3.2 "Hardware Control Protocol". The hardware controller also creates the camera class. The camera class opens a video stream and fills a ring buffer continuously with images of the created video stream. Those images can also be requested by the state machine by the hard-

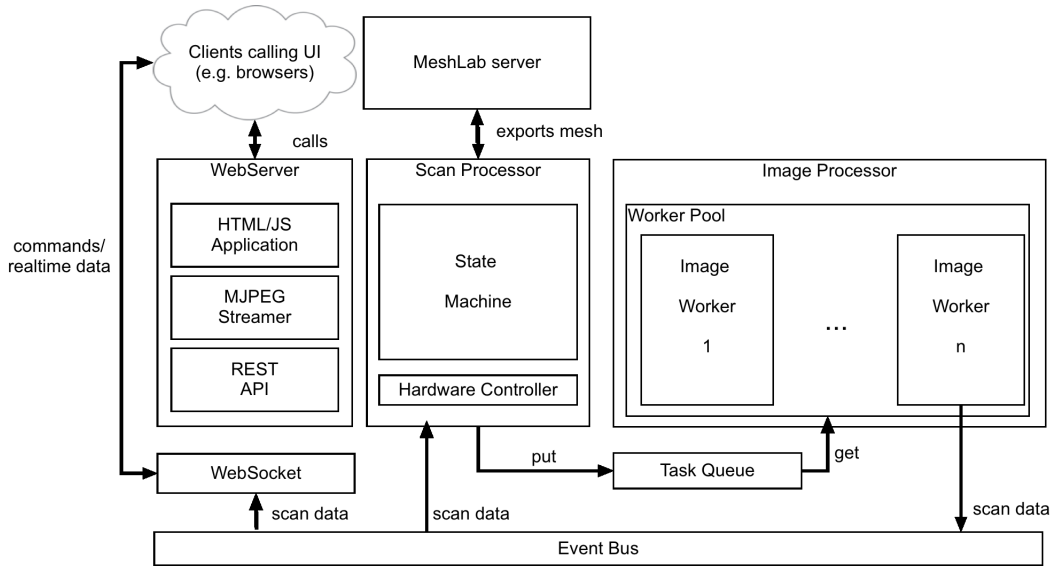

Figure 3.7: FabScan Pi server back-end architecture

ware controller.

The scan state of the state machine first starts the color and texture scan. The scan processor calculates the number of images depending on the scan resolution. Afterwards the processor takes as many pictures as it calculated before. The object scan step is similar with the difference that the laser is switched on during the whole scan process. Every picture is packed into an image task and added to the task queue.

Another module is the image processor. The image processor contains a worker pool and creates image workers in the pool. The worker pool consists of n image workers. The number of image workers depends on the number of CPU cores. In case of the Raspberry Pi 2, four image workers are created. Each image worker has access to the task queue. An image worker runs a loop which checks if a new image task is available. When a new task is available

the image worker takes the task out of the queue and starts processing the tasks data. In case of a color image the image worker saves the image with a reference number to the Raspberry Pi's SD-card. If the task contains a laser image, the image worker starts the triangulation process. At the end of this process the corresponding color image is loaded again to find the colors of the triangulated points. The result is sent to the WebSocket module and back to the state machine by using the event bus. The WebSocket connection sends the new point data to the client. The points are also added to the point cloud instance in the scan processor. Finally, the scan processor indicates the end of a full turn and with it the completion of a full scan. The points are written to a PLY file and the scanning post process is triggered.

MeshLab Server is used for exporting X3D and STL files.

The post process calls MeshLab Server to convert the PLY data into different printable mesh formats. A MeshLab Server batch can be configured by MLX filter scripts. Those filter scripts contain a list of export, cleaning and meshing filters which are applied to the input file. Afterwards the exported files are saved to the FabScan Pi output folder. A further message is send through the WebSocket connection to inform the user interface that the scan process is complete. The following listing shows an example of an MLX filter script which creates a watertight mesh.

```
1  <!DOCTYPE FilterScript>
2  <FilterScript>
3   <filter name="Smooths␣normals␣on␣a␣point␣sets">
4    <Param type="RichInt" value="20" name="K"/>
5    <Param type="RichBool" value="false" name="useDist"/>
6   </filter>
7   <filter name="Surface␣Reconstruction:␣Poisson">
8    <Param type="RichInt" value="6" name="OctDepth"/>
9    <Param type="RichInt" value="6" name="SolverDivide"/>
10   <Param type="RichFloat" value="1" name="SamplesPerNode"/>
11   <Param type="RichFloat" value="1" name="Offset"/>
12  </filter>
13 </FilterScript>
```

Figure 3.8 shows the results of the MeshLab post processing chain. For the process in the image a duck was scanned with medium resolution. First a filter for meshing the point cloud to a meshed PLY was applied. The second MLX filter used a Poison Surface Reconstruction algorithm to get a watertight mesh. At the end an additional export filter

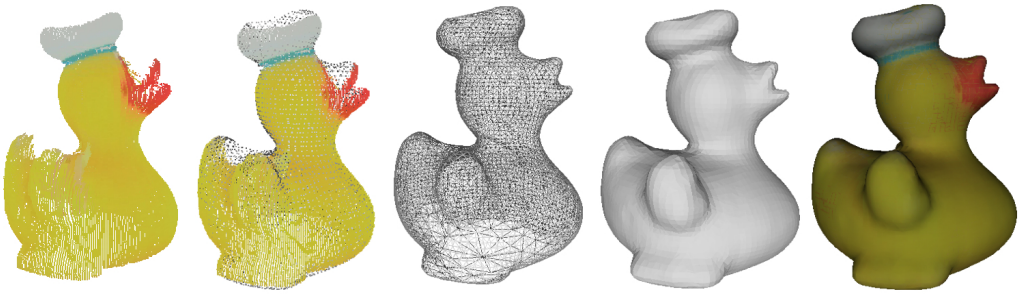

Figure 3.8: From left to right. First: point cloud output of FabScan Pi. Second: PLY with mesh reconstruction output of MeshLab, containing colors. Third: meshed STL in mesh view. Fourth: STL with surface, export by MeshLab. Fifth: Exported X3D file with full texture information by MeshLab filter.

was used to create a textured mesh of the point cloud. The textured mesh can be used to print a colored model.

The WebServer module is used to serve the HTML and JavaScript files to the client. Also the REST API is encapsulated by the WebServer. Another task of the WebServer is to provide an MJPEG camera stream for the live settings dialog which is described in 3.3.4 “Functionality”.

3.3.4 Web-enabled User Interface

FabScan Pi provides a web-enabled user interface. The interface is an application which is written in JavaScript with the usage of AngularJS. It is widely independent of the FabScan back-end server. For the graphical visualization the user interface uses the WebGL standard. To keep things simple, the FabScan Pi user interface also uses a JavaScript 3D library called ThreeJS[1]. This section starts with the a mockup of the web-enabled user interface. The second part of this section describes the functionality and different features of the final web-enabled FabScan Pi user interface.

[1] http://threejs.org

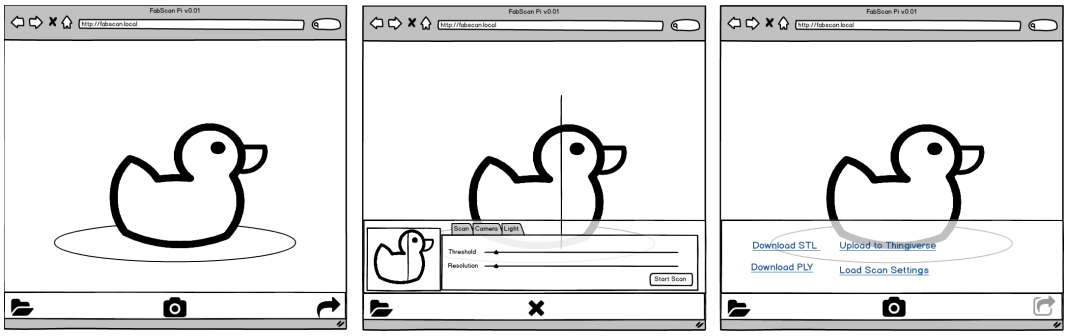

Figure 3.9: Mock-up of the web-enabled user interface. Right: Main View, Middle: Settings View, Right: Share View

User Interface Mockup

The first version of the FabScan Pi web-enabled user interface was created with a mockup software tool called balsamiq[2]. The user interface of the old FabScan software was largely inspired by Apple's Photo Booth[3] [Eng11]. For keeping the users experience the main concept of the old user interface was adopted (Figure 3.9 Left). But because FabScan Pi is web-enabled, some concepts have to be changed. By choosing a responsive design approach the web-enabled user interface should be usable on mobile devices like tablets and smart phones.

Users in the FabScan group mentioned that it was troublesome to use the settings dialog in the old FabScan software. This is one of the shortcomings described in 2.5 "Shortcomings of the FabScan CUBE". Users of the old FabScan software have to perform a full scan to check if the scan quality is improved by the updted settings. With an average scan time of 15 minutes for a scan in medium quality, a lot of time is lost to find the best scanner settings. Since the scan quality is largely depending on the laser line recognition and ambient light, a settings dialog is displayed before each scan.

The settings dialog provides a live preview of the laser line recognition (Figure 3.9 Middle). Every time the user

[2]https://balsamiq.com
[3]https://www.apple.com/osx/apps/#photobooth

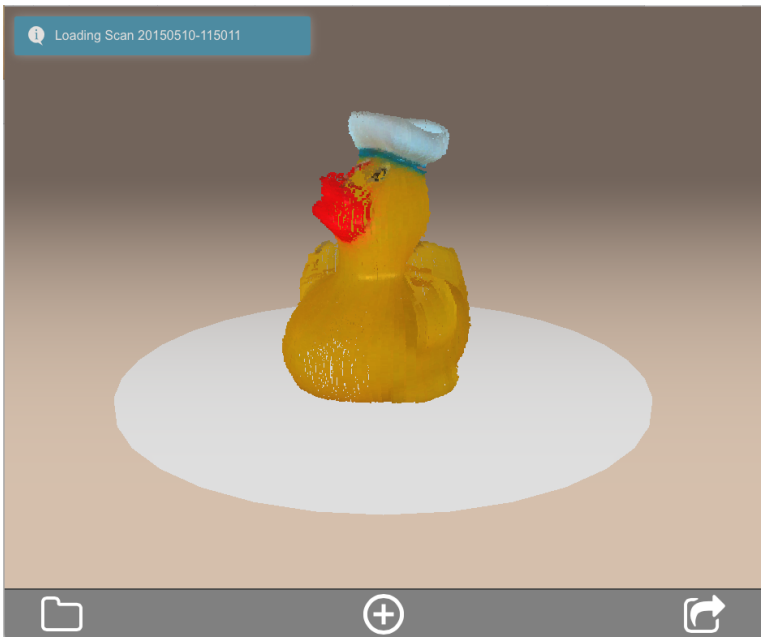

Figure 3.10: Main view. The view which is shown when fabscan.local is called in a browser.

changes the scan, camera or light settings the changes are visible in a real-time video stream. Also the resolution is set in the settings dialog. Those settings are saved in a file beside the other scan data.

When the scan process is completed a share button is shown. The share button opens the share dialog (Figure 3.9 Right). In the share dialog the user can chose different options to download or share the scanned data. The share dialog also offers the possibility of loading the scan settings which were used for the related object. This can be helpful when the user wants to scan an object with similar shape, color or surface properties.

Functionality

The main view of the FabScan Pi web-enabled user interface is shown in Figure 3.10. In the main view the user is able to call three different dialogs. A dialog opens by click-

Figure 3.11: Loading dialog. Offers the opportunity to load recent scans.

ing the icons shown at the bottom of the main view.

The folder icon calls the loading dialog. The loading dialog is also inspired by Apple's Photo Booth. It provides a gallery where the user can slide through previous scans. A small label on the gallery preview image shows some scan related meta data like the date of the scan. The user can explore the gallery by clicking on arrows on both sides. A scan can be loaded by clicking on the preview image. If the user clicks on a preview image the loading dialog closes the scan is loaded in the main window and the share icon is shown.

When the user presses the share button, the share dialog opens (Figure 3.12). The share dialog offers different file operations including several download options. The user can download the scanned object in PLY, STL or X3D format. Also, the user is able to load the settings from prior scanned objects. This is a useful feature if the user wants to scan an object with similar properties as a previous scanned object.

Figure 3.12: Sharing dialog. Offers the opportunity to download scans in different formats and delete scans.

The settings dialog of the FabScan Pi user interface shows a live video stream on the left side. The video stream contains a live-view of the current laser line recognition algorithm. If the user changes the settings sliders on the right side of the dialog, the laser line recognition changes in real-time. The recognized laser line is shown by a blue line in the live video stream. Best scan results are achieved when the blue laser line and the ”real” red laser line match exactly.

The user has the choice between three different setting tabs. The first tab includes all scan relevant settings like resolution and threshold values. The meaning of the threshold value is described in the FabScan bachelor thesis [Eng11]. The second tab includes camera settings like contrast, hue, saturation and brightness. The last tab contains the ambient light settings. With the sliders in the third tab the user is able to control the lighting in the inside of the FabScan Pi.

The user can start the scan out of the settings dialog by clicking the scan button. Afterwards the settings dialog

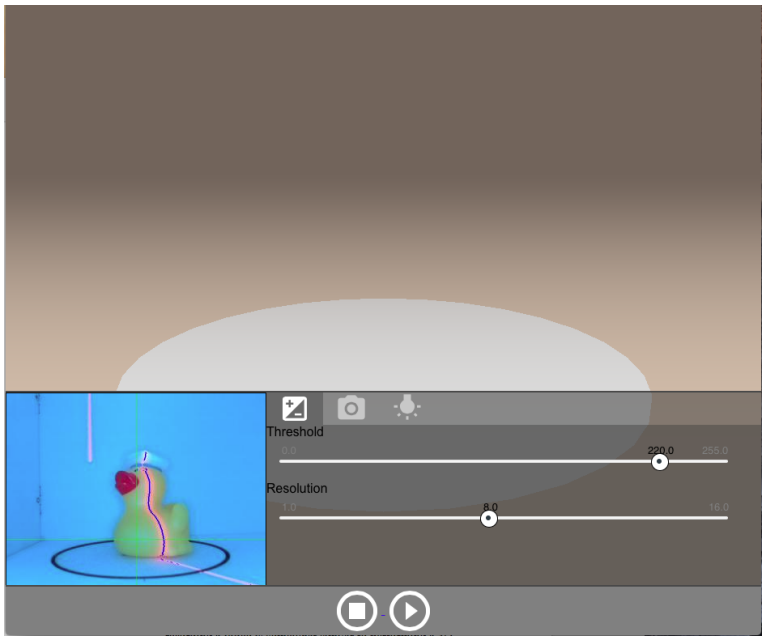

Figure 3.13: Settings dialog. Offers the opportunity to find the best scan settings. Left side of dialog: live laser line recognition. Right side of dialog: sliders for changing settings.

is closed and the scan process starts. The points for the current object position are added to the view as they are scanned. After a successful scan, the share icon for the share dialog appears at the bottom left side of the user interface.

Chapter 4

Evaluation

4.1 Requirements

In this section, it is discussed whether and to what degree the requirements from 3.1 “Requirements” are met.

4.1.1 Affordability of FabScan Pi

In this paragraph the cost of the final FabScan Pi hardware prototype from chapter 3.2.3 “Final Hardware Prototype” is calculated. The price of the FabScan Pi setup is compared to the price of the FabScan CUBE setup in Table 4.1.

Table 4.1 shows that the price difference between FabScan Pi setup and FabScan CUBE setup is about 30 Euro, which makes the FabScan Pi almost as affordable as the FabScan CUBE, but with the benefit of a small computer device what makes the FabScan Pi a stand-alone and web-enabled device.

Component	Count	FabScan Pi Price	FabScan CUBE Price
Stepper motor	2	15,88 €	15,88 €
Motor driver	2	9,95 €	7,96 €
Laser cut parts + screws	1	35,00 €	35,00 €
Power supply (12v / 5v)	1	14,95 €	14,95 €
Laser module	1	5,90 €	5,90 €
Arduino	1	-	23,80 €
FabScan shield	1	-	10,00 €
Logitech WebCam	1	-	22,95 €
12 RGB LED Ring	1	7,79 €	-
FabScan Pi shield	1	20,00 €	-
Raspberry Pi	1	38,95 €	-
Raspberry Pi camera	1	20,00 €	-
	Total	168.42 €	136.44 €

Table 4.1: FabScan Affordability

4.1.2 Do-it-Yourself

Even the FabScan Pi can be build by everyone. The hardware setup changed in a few details. The Arduino and FabScan shield is replaced by the FabScan Pi shield and the Raspberry Pi 2. The webcam is replaced by the Raspberry Pi camera module (see 3.2.1 “First Hardware Prototype”). Due to the fact that the Raspberry Pi camera module is delivered as a small PCB, it is not necessary anymore to disassemble a webcam module. The software installation process is also easier with the FabScan Pi. The user only needs to download and flash the SD-Card image. The firmware is installed automatically with the first start of the FabScan Pi server.

4.1.3 Stand-alone Device

By using a Raspberry Pi 2 with the FabScan Pi server software and the web-enabled user interface, there is no need to install software on third party hardware like computers, laptops, etc. The FabScan Pi is a full usable stand-alone 3D scanner.

4.1.4 Web-enabled User Interface

In chapter 3.3.4 “Web-enabled User Interface” the development process of the web-enabled user interface is described. With the web-enabled user interface the user is able to perform a scan from any device which supports an HTML5 compatible web-browser with WebGL support.

4.1.5 Integration of Meshlab

The FabScan Pi back-end server calls the Meshlab command line client for the meshing post process. With this mechanism the user is able to decide what file format he wants to export. The user is able to write new Meshlab filter chains as described in 3.3.3 “Back-end server”.

4.1.6 Downloading Scans in Different Formats

The FabScan Pi’s web-enabled user interface provides the opportunity of downloading the scans in different file formats. The user is able to download scans in STL, PLY and X3D.

4.1.7 Provide Settings Preview

The FasbScan Pi’s web-enabled user interface provides a live settings dialog (see 3.3.4 “Web-enabled User Interface”). In this dialog the user can change the scan settings and he gets a direct feedback of the current laser line recognition.

4.1.8 Save and Load Scan Settings

Scan settings made in the settings dialog are saved with the scan result. The user is able to load those scan settings, re-

lated to the scanned object, by using the share dialog. With this feature the user can scan objects with similar properties without the effort of finding the best scan settings in the settings dialog.

4.1.9 Improving Scan Quality

A new feqautre of the FabScan Pi hardware is an RGB LED ring that increases the quality of the scan results. With a white light during a color scan the colors are brighter and clearer. The second quality improvement was reached by introducing the live settings dialog with the web-enabled user interface. The settings dialog in the web-enabled user interface helps the user to get the best results without much configuration effort. The user can find the best laser line recognition by a live video stream which shows the recognized laser line.

4.1.10 Improving Scan Speed

The FabScan Pi back-end server supports multi core operations that divide the scan process into multiple image calculation tasks. This process increases the speed of the FabScan software. A scan in medium quality with the FabScan CUBE software on a PC takes about 15 minutes. The same object scanned with the FabScan Pi software on a Raspberry Pi takes only about 3 minutes. The scan speed is increased by a factor of 5.

4.1.11 Usability

The web-enabled interface is kept as simple as possible. With FabScan Pi the user is able to perform a scan with a minimum number of clicks. With the new settings dialog the user can get the best scan result without repeating the scan process for each scanner settings change.

Chapter 5

Summary and future work

In the previous chapters the idea and the development process of the FabScan as a web-enabled stand-alone 3D Scanner were presented. At last, this chapter will give a summary of the most important aspects and the possible insights into the future.

5.1 Summary and contributions

In this work, an open-source, web-enabled, stand-alone 3D scanner was created. As basis of my work the FabScan CUBE was used. The proposals of the FabScan Google user group were examined and the shortcomings of the FabScan CUBE were determined. With the determined shortcomings and the comparison of current available 3D scanners the requirements for the FabScan Pi system were formed. After several hardware prototypes some of the issues of the FabScan CUBE were improved and a stand-alone device was made out of the FabScan CUBE. With the development of the FabScan Pi software, the FabScan turned into a web-enabled 3D scanner.

My contribution to the community is a new version of the

FabScan 3D scanner. The new scanner is with about 160 €nearby as affordable as the FabScan CUBE. The new software offers several advantages, like improvements of usability, scan speed and quality.

5.2 Future work

In this chapter some ideas and suggestions which came up during the process of the thesis are presented. As this thesis concentrates on the shortcomings and user group suggestions of the FabScan 3D scanner, the following states potential further improvements of the FabScan Pi.

5.2.1 Adding more REST API functions

Currently the FabScan Pi back-end server provides only basic REST API functionality. In a new version it can be useful if the REST API supports also command functions. With command functions in the REST API the FabScan Pi back-end can be controlled through third party web applications.

5.2.2 Introducing ICP algorithm

One aspect of laser range scanners is that most objects with re-entrant angles are difficult to digitize. The laser can not reach those angles or the camera is not able to capture parts of the object where the laser is covered by other object parts.

The ICP (Iterative Closest Point) algorithm can solve this problem. ICP is used to match point clouds. With the implementation of ICP it would be possible to scan the object with different positions. Several point clouds of different object positions could be matched to one point cloud. With this feature the re-entrant angle problem can be solved.

5.2.3 Configuration of Meshlab in user interface

In FabScan Pi Meshlab was introduced. The current FabScan Pi software does not provide an upload or selection mode for custom filter scripts. It would be better if the user can create and upload custom created Meshlab filter scripts. For this purpose a drop down selection box, for selecting the export filters, has to be added to the settings dialog.

5.2.4 Scanner auto calibration

During the work on this thesis several FabScan users online and at Maker Faires mentioned their FabScan user experiences. It became clear that the FabScan needs an auto calibration mechanism. An auto calibration mechanism can be realized by placing a checkerboard on the turntable and measuring the distances of camera and turntable. Also the horizontal and vertical alignment of the camera can be determined by the measurement mechanism.

5.2.5 Auto settings dialog

The usability could be improved by introducing an auto settings dialog. If the FabScan is able to detect the object material and color, it would be possible to detect the best matching scanner settings. With an auto setting feature maybe the scan process speed and quality can be increased.

5.2.6 Adding Octoprint support

Octoprint is a print server for 3D printers which is runnable on a Raspberry Pi. Octoprint support in the FabScan Pi web-enabled user interface would enable the user to use the FabScan Pi as a copy device. Octoprint supports a REST API with an extensive feature set. Through a REST API an interface for the FabScan Pi to Octoprint is easy to develop.

Appendix A

Raspberry Pi HAT 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.

File: RPI-FabScan-HAT.brd[a]

[a] https://github.com/watterott/RPi-FabScan-HAT/RPI-FabScan-HAT.brd

File: RPI-FabScan-HAT.sch[a]

[a] https://github.com/watterott/RPi-FabScan-HAT/RPI-FabScan-HAT.sch

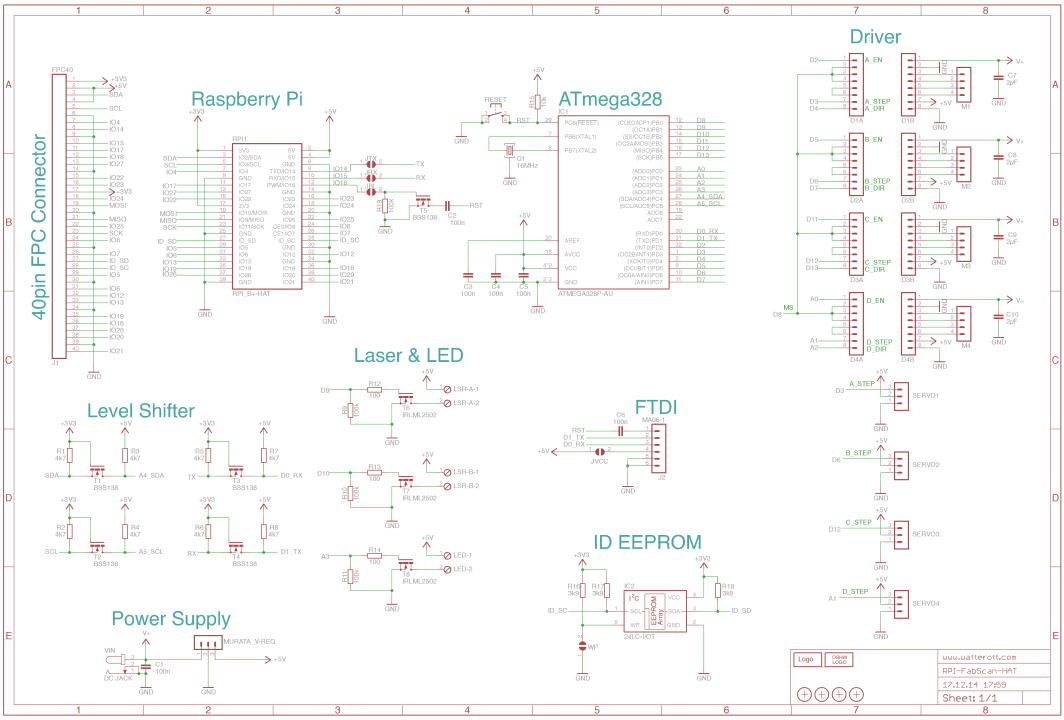

Figure A.1: FabScan Pi HAT schematics

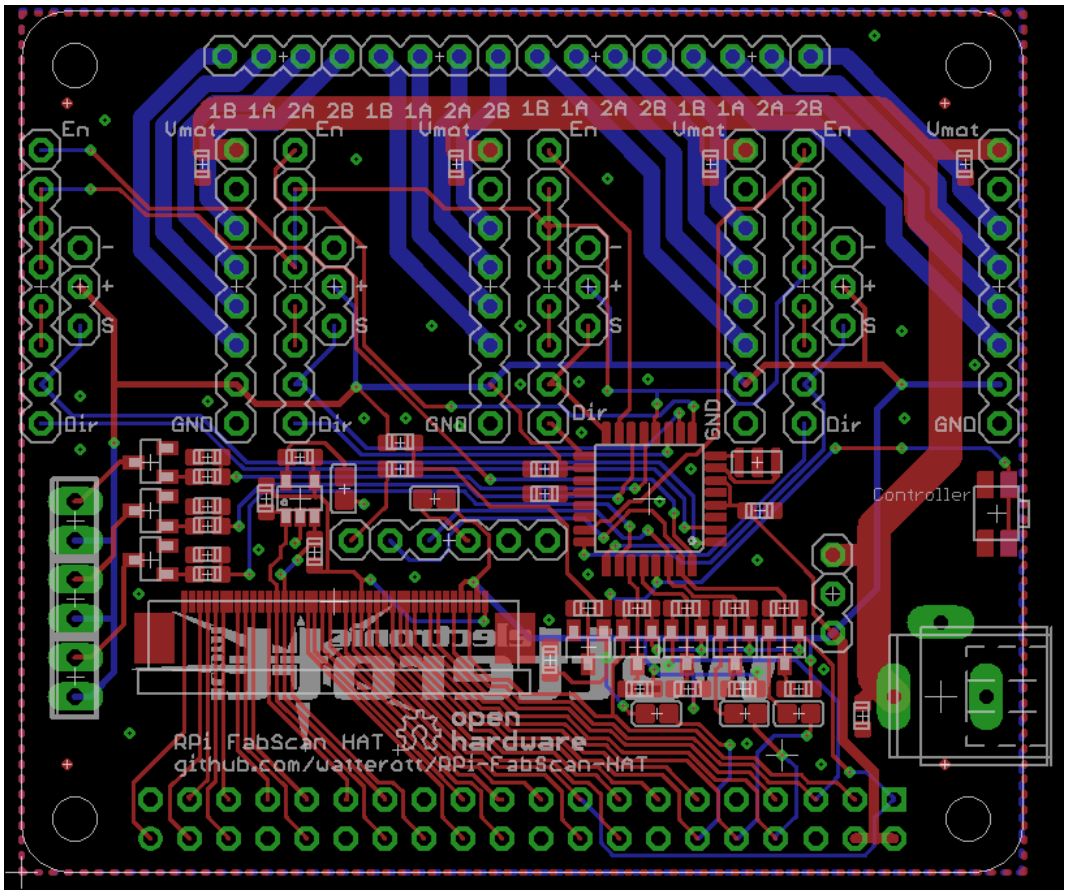

Figure A.2: FabScan Pi HAT PCB layout

Appendix B

Schematics for the Laser Cutter Parts

The laser cutable parts are provided under the following download link. The files are updated when there is a new version available.

File: picam_mount.dxf[a]

[a] https://github.com/mariolukas/FabScanPi-LaserCutParts

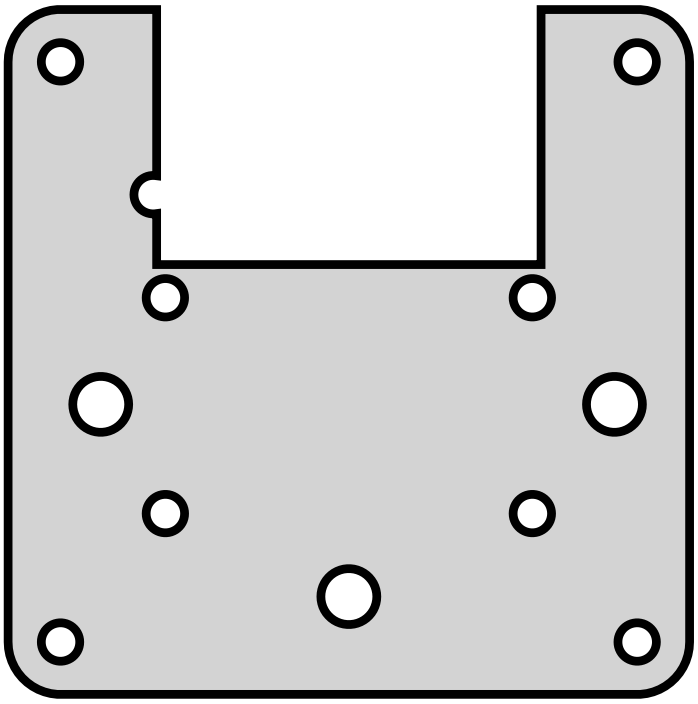

Figure B.1: FabScan camera mount

Appendix C

Source Code

The source code of the back-end and the web-enabled user interface can be found under the following download link. The code can also be found on GitHub.

File: FabScan Pi Frontend[a]

[a] https://github.com/mariolukas/FabScanPi-Frontend

File: FabScan Pi Backend[a]

[a] https://github.com/mariolukas/FabScanPi-Backend

Bibliography

[Ada14] James Adams. Definition of raspberry pi hat. https://www.raspberrypi.org/introducing-raspberry-pi-hats/, 2014. Accessed: 2015-05-01.

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

[Bor01] Jan Borchers. *A Pattern Approach to Interaction Design*. 2001.

[Bou15] Paul Bourke. Ply - polygon file format. http://paulbourke.net/dataformats/ply/, June 2015. Accessed: 2015-06-02.

[Bur89] Marshall Burns. The stl format. http://www.fabbers.com/tech/STL_Format, October 1989. Accessed: 2015-06-02.

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

[Eng11] Francis Engelmann. Fabscan affordable 3d laser scanning of physical objects. Bachelor thesis, RWTH Aachen, The Media Computing Group, 2011.

[IF11] A. Melnikov I. Fette. The websocket protocol. https://tools.ietf.org/html/rfc6455, December 2011. Accessed: 2015-06-01.

[Inc06] On-Net Surveillance Systems Inc. Mjpeg vs mpeg4 - understanding the differences, advantages and disadvantages of each compression technique. http://www.onssi.

com/downloads/OnSSI_WP_compression_techniques.pdf, 2006. Accessed: 2015-05-05.

[Jon14] Dave Jones. picamera - documentation of the python raspberry pi camera interface. https://picamera.readthedocs.org/en/release-1.10/fov.html, 2014. Accessed: 2015-04-22.

[Kaa13] Corporation Kaazing. What is websocket? http://www.websocket.org, 2013. Accessed: 2015-06-10.

[LT09] Douglas Lanman and Gabriel Taubin. Build your own 3d scanner: 3d photography for beginners. August 2009.

[Mes00] Thomas R. Kramer; Frederick M. Proctor; Elena R. Messina;. *The NIST RS274NGC Interpreter - Version 3*. 2000.

[pyt15] python.org python. What is python? executive summery. https://www.python.org/doc/essays/blurb/, 2015. Accessed: 2015-06-29.

[Rou12] Margaret Rouse. Hardware glossary - definition: Raspberry pi. http://whatis.techtarget.com/definition/Raspberry-Pi-35-computer, April 2012. Accessed: 2015-06-30.

[Vuk07] Vlad Vukicevic. Canvas 3d historical. https://wiki.mozilla.org/Canvas:3D/Historical, April 2007. Accessed: 2015-04-27.

[Web15] Consortium Web3D. What is x3d. http://www.web3d.org/x3d/what-x3d, October 2015. Accessed: 2015-06-02.

Index

Typeset July 7, 2015