Pavel Ivanov

Three-dimensional Scanning of Objects Using a Mobile Phone

Photogrammetry Silhouette Technique

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Engineering

Thesis

March 21, 2017

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Pavel Ivanov Three-dimensional scanning of objects using a mobile phone 65 pages 21 March 2017
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	
Instructor	Harri Airaksinen, Principal Lecturer

The purpose of this study was to find a solution and develop a working mobile phone application that allows user scanning of specific objects with usage of standard phone configuration without involvement of any additional external devices. This thesis demonstrates the implementation of the 3D scanning application including problems found during development and results of the application workflow.

An appropriate solution was found in the form of silhouette scanning technique. The solution allows reconstructing the model with moderate inaccuracies using a combination of data gathered from multiple embedded sensors and processing of the data.

A semi-successful mobile phone application was built in the project by using an open-course code library OpenCV for the Google Android mobile platform. The application produces a 3D model billet of the scanned object on the basis of taken images and rotation angles, yet an accurate version of the object cannot be recreated and further work is needed before the application will reach a sustainable and competitive state.

Keywords	3D scanning, photogrammetry, silhouette, mobile phone, position sensors

Contents

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
CAD	Computer-aided design
CGI	Computer generated imagery
DSLR	Digital single-lens reflex
IT	Information technology
GUI	Graphical user interface
MMI	Man-to-machine interface
NURBS	Non uniform Rational B-Spline
OBJ	File format used for storing data about 3D model
OpenCV	Open library of computer vision functions

1 Introduction

Throughout human history people have endeavoured to cease the time, seize the moment and embody it in any available form of fine arts. Humans have devised and nurtured a multitude of arts, starting from the most ancient cave drawings and Venus figurines, statues of Ancient Greece and the Renaissance period, and including the relatively recent epoch with the involvement of the latest chemical, mechanical and electrical accomplishments. Artists, sculptors and artisans have spent years of their lives in the recreation of only one moment or idea. Over time, progress has brought and still brings new abilities and opportunities for realization of ideas to creative people.

Since the middle of the twentieth century, computer technologies started development with a quick pace. Breakthroughs in chemistry have permitted minimizing significantly electronic components needed for production of computer hardware. The consequence of this was the appearance of first mobile phones and the rapid development of their capabilities and rising amount of embedded technologies. In twenty years, phones have transformed from simple transportable communication devices to miniature computers with a single camera or several cameras and various sensors, including accelerometer, magnetometer and gyroscope.

The invention of the computer presented an opportunity to humanity to develop a new art - computer graphics. The development of computer graphics advanced similarly to the development of human arts: from the simplest textual and pixel graphics to photorealistic three-dimensional models, from necessity to calculate every value and draw every pixel manually to a dynamic generation of world models and drawings with the help of machine learning and artificial intelligence.

Collaboratively with three-dimensional modelling, three-dimensional scanning is evolving, drastically behind the modelling by means of time and broadness of used technologies. The scanning allows recreating a billet of a meaningful model for subsequent refinement. However, present realizations of the scanning process do not allow reconstructing top-quality models without a participation of a specialist and require considerable financial and time expenditures for procurement and maintenance of applied high technology equipment. In addition, the mobile application market does not have a free, open source and commonly available solution on a turnkey basis without the involvement of supplementary devices or protracted configuration and calibration.

The objective of the study described in this thesis was a creation of a fully functioning three-dimensional scanning application. The application was developed for Android-based mobile devices of a standard configuration and used the open source library OpenCV, which is mainly aimed at image processing and real-time computer vision. The scanning application covers all working processes, starting from taking a camera shot to recreation of a model in the form of an OBJ file, which is a widely-used standard in the 3D modelling industry and suitable for the subsequent import and modification.

2 Brief history of computer graphics

2.1 Origins

The “computer graphics” term was used for the first time publicly in the year 1960 by the art director of The Boeing Company, William Fetter, who worked with research of graphic technologies for a future application in the development of jet planes drafts [1]. Multiple sources also state him saying that his work colleague Verne Hudson suggested it to him originally [2]. At first, the term computer graphics was used inside the company for description of works and drafts made by a mechanical plotter controlled by a computer [1].

Fetter was convinced that the appearance and wide distribution of computers will change and broaden up the creative potential of humanity and that the computer in its complete form will be serving as a tool for an explicit projection of thoughts and emotions. Together with his colleagues, programmers and engineers of Boeing Company, Fetter developed first in the world a computer model of the human and its motion simulations, shown in figure **1**, thereby demonstrating a revolutionary new face of computer facilities application. His following artistic work and participation in the Arts and Technology movement inspired him to support a growing collaborative society of artists and engineers. His efforts to build a bridge of communication between artists and engineers led to the creation of “Circles I”, an early computer-generated film made in cooperation between Boeing engineers, Doris Chase, local filmmakers and artists. Fetter’s dedication and endeavours showed to artists a computer as an instrument to work with and gave a first impetus to a wide distribution of computer graphics. [1.]

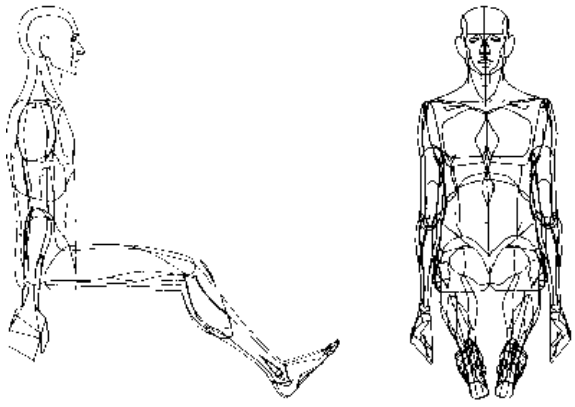


Figure 1. William Fetters Boeing man model. Reprinted from cgsociety.com [3].

2.2 3D modelling and animation

In 1972, two graduate students, Ed Catmull and Fred Parke, of Utah University introduced their thesis project – a one-minute movie with a three-dimensional polygonal representation of a human hand, which became one of the first prototypes of 3D computer animation. The movie contains a process of the creation and several short computer animations – a hand rotation, a fingers' flexure, pointing by a finger to a watcher and a hand's wireframe model at the end. [4.] Selected movie frames are shown in figure **2**.

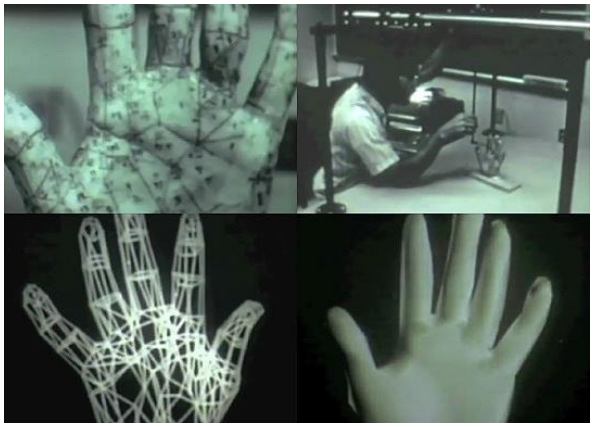


Figure 2. Frames from 'A Computer Animated Hand'. Copied from Steemit [5].

Later, Catmull became one of the cofounders of a company called Pixar, specializing in the production of feature-length movies fully made with the computer graphics [4].

In 1977, the first movie of the George Lucas' Star Wars trilogy was released in wide screen theaters and it had a three-dimensional wireframe imagery that represented the Death Star inner plans and structure, shown in figure **3**. It was made by a computer artist Larry Cuba on the basis of a photo of a matte painting by using a minicomputer with a monochrome display and a vector-graphics system that allowed drawing only lines. [6.]

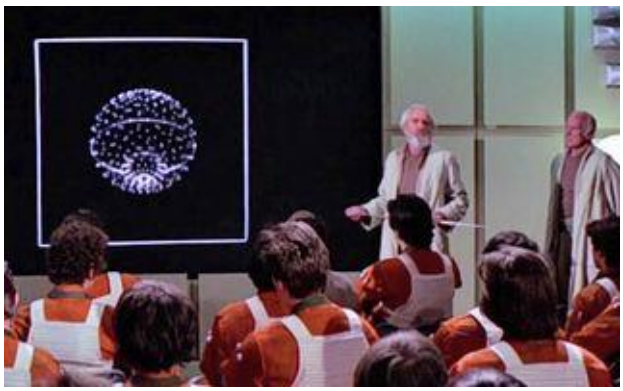


Figure 3. Frame from ‘Star Wars’ movie. Reprinted from Price [6].

The computer-generated imagery started to appear in high-budget movies during the 1980’s and the 1990’s, the most remarkable and well-known being Tron, Terminator 2, Start Trek IV and James Cameron’s The Abyss. In 1995, Ed Catmull and his company Pixar released the feature length movie Toy Story made in full of the 3D computer-generated imagery, which was nominated and won multiple awards for the animation, technical and special achievements and innovations [7]. Pixar and a number of other companies, including Disney and DreamWorks, has developed the idea of a computer-generated imagery feature length cinematography and gave to the world such marvelous movies as WALL-E, Zootopia, Finding Nemo, Shrek. [8.]

Creation of CGI-based movies is no longer a prerogative for a dedicated business. Motion picture companies and directors are able to recruit independent graphic studios for implementation of particular aspects or parts of the movie. In 2009, James Cameron concluded a contract with Weta Digital and Industrial Light and Magic companies to accomplish work on visual effects, modelling and a motion capture of actors for the creation of the movie Avatar. Production also involved such companies as Microsoft, which developed and provided a cloud computing technology and the Digital Asset Management system specifically for the movie, a server render farm of Hewlett-Packard servers and Pixar’s technologies of Renderman and Alfred queue management system. [9.] A significant speedup of the work process and the release of the incredibly high-quality CGI in the Avatar movie were the result of this extraordinary union. The movie won all possible nominations related to computer graphics including one of the most popular – the Oscars [10].

Nowadays the computer animation can be seen anywhere, as special effects on television, in cinemas and games, and in different multimedia applications. It keeps evolving, trying to reach the most photorealistic level of the computer graphics. The achievement of the photorealism will heavily affect the production processes in the film industry and the life of ordinary people.

2.3 3D modelling software

The first commercial software for creation and working with the 3D graphics were launched in the 1980’s. A well-known company called Autodesk entered the market in

1982 with its first and primary product AutoCAD, a commercial drafting software application for computer-aided design, which allowed and simplified the process of the draft production on personal computers. Possibility to create three-dimensional models in the manner of either transformation of two-dimensional drafts or a model creation from a scratch was added in subsequent versions of the application. The company cultivated the idea of working with the 3D graphics and in 1990 released a fully dedicated modelling software called 3D Studio, known as 3DS Max nowadays, for the DOS operational system. [11.]

Alongside with Autodesk, competitive companies were developing and fighting for the market of computer graphics. Wavefront, Softimage and Side Effects Software were the most notable of them.

The main products of Softimage, Softimage 3D and its successor XSI, were aimed to work with CGI for feature length movies. They were used for the creation of visual effects of Titanic, Fifth Element, 300 and Jurassic Park. During its lifespan, Softimage heavily influenced the animation industry. Softimage was the first to bring animation tools to personal computers under the Windows system and as a result increased affordability and accessibility of tools to end customers. Softimage developed the first integrated animation, effects and post-production systems. In 1994, Microsoft bought the company to strengthen its own role in the multimedia and interactive television production, yet six years later sold Softimage to Avid Technologies, which wanted to expand its own business to the CGI market. [12.]

Wavefront products were oriented to cover a wide range of computer graphics applications and were used on television, in movies, and in academic and engineering visualizations. Wavefront started its way with release of Advanced Visualizer, a suite of independent applications for an image and graphic animation processing that was used for the production of CGI in Jurassic Park, Terminator 2 and Abyss. Wavefront was the first in the world to release one of the earliest commercial tools for scientific visualizations, named Data Visualizer. In 1995, the Wavefront company was bought by Silicon Graphics Inc. and was merged with Alias Systems Corporation into the super company called Alias|Wavefront as a response to Microsoft's attempt of taking over the 3D market by purchasing the Softimage company. Alias|Wavefront released their most well-known product, a 3D modelling and animation application called Maya, based on the code of Advanced Visualizer. [13.]

Autodesk company was more prosperous than other companies in the industry. It avoided being purchased by big IT corporations and in the years of 2006 and 2008 it was able to buy its previous competitor Softimage from Avid [14] and Alias|Wavefront from Silicon Graphics Inc. [15], becoming one of the largest companies in the industry. Its major role in the market permits the Autodesk company to increase the prices of its own products [16] and implement a subscription-only based purchase model as a result [17], neglecting extremely negative responses from customers.

The modelling software market was relatively closed until the end of the 1990's. Small companies with their own products stand almost no chance in competition with industry leaders. Nonetheless, an inaccessible affordability for private customers, a high barrier to entry and an absence of a wide selection on the market stimulated the appearance of a number of free alternatives, the most notable being Blender, Anim8or and Open CASCADE. Blender started as an in-home toolset, made by Ton Roosendaal, and later was transformed into a commercial product under Ton's company named Not a Number. The company used to provide services and commercial products for the Blender and was financed by several investment companies. Low sales and economic problems made investors to discontinue financing in early 2002, which led to open sourcing Blender under the GNU license and the community support and development. [18.]

Blender was considered as the most popular modelling software for 3D printing in 2015. Nevertheless, as it is shown in figure **4**, the number of Autodesk products (AutoCAD, Maya, 3DS Max, 123D Design, Inventor, Fusion 360, and Meshmixer) allows naming Autodesk company as the largest player in the industry with the widest audience coverage. [19.]

Top 25

Most Popular 3D Modeling Software for 3D Printing

		General		3D Printing Community				Total
		Social	Website	Forums	YouTube	Databases	Google	Score
1	Blender	61	91	100	100	27	100	**80**
2	SketchUP	87	82	79	49	80	74	**75**
3	SolidWorks	95	81	42	52	25	75	**62**
4	AutoCAD	100	78	46	43	4	85	**59**
5	Maya	91	80	35	50	3	93	**59**
6	3DS Max	90	83	24	53	2	78	**55**
7	Inventor	98	80	29	31	15	75	**55**
8	Tinkercad	78	57	38	5	100	31	**51**
9	ZBrush	83	69	45	42	4	50	**49**
10	Cinema 4D	84	76	6	28	1	62	**43**
11	123D Design	85	67	21	14	18	50	**42**
12	OpenSCAD	1	65	33	2	100	29	**38**
13	Rhinoceros	17	75	50	21	6	49	**36**
14	Modo	82	63	10	9	1	45	**35**
15	Fusion 360	93	81	10	3	2	4	**32**
16	Meshmixer	1	62	18	7	9	28	**21**
17	LightWave	23	52	1	8	0	32	**19**
18	Sculptris	0	67	7	6	4	26	**19**
19	Grasshopper	9	60	4	5	1	32	**18**
20	FreeCAD	4	59	15	8	11	5	**17**
21	MoI3D	0	53	3	1	0	28	**14**
22	3Dtin	4	57	0	0	11	1	**12**
23	Wings3D	0	66	1	1	0	2	**12**
24	K-3D	0	62	1	1	0	2	**11**
25	BRL-CAD	0	60	1	0	0	1	**11**

i.materialise
i.materialise.com

Figure 4. Popular modelling software used for 3D printing. Copied from Top 25: Most Popular 3D Modeling & Design Software for 3D Printing [19].

2.4 3D scanners

The first attempt to create a functional 3D scanner was taken in 1980's. It was a contact probe scanner, which could provide precise results but the complete scanning process took a very long time, and the scanner could not scan objects with a soft surface well. In search of speeding up the scanning process and avoiding the prodding problems of objects, researchers decided to look into other possibilities and started developing scanning based on optical technologies. [20.]

At the time, only three types of the optical sensors were available, shown in figure **5**:

- Point sensor, which scanned only one point of the object at once and due to this did a lot of physical movement across the scanning object [20];
- Area sensor, which made an area scanning at once, but was technically difficult to realize [20];
- Stripe sensor, which captured a band of multiple points of object at once [20].

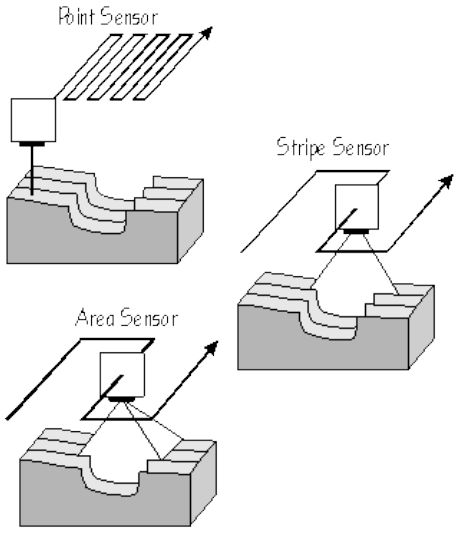


Figure 5. Point, Area and Stripe scanning optical technologies. Reprinted from Hoffmann [20].

Stripe technology was the golden mean between the capturing speed, accuracy and the complexity of technical realization and was chosen for further development. The optical sensor needed to accomplish the scanning multiple times from different positions for capturing objects in three-dimensional space. Data, collected from scans, could be represented by millions and millions points and had duplications, which raised difficulties to software for processing this kind of data amount. [20.]

In the 1980's, Cyberware Laboratories developed the Head Scanner, the first scanner to capture human shapes and proportions for the animation industry. It used a low intensity laser to create a highlighted profile of an object after which a reading sensor captured the profile. In an extended version of the scanner, a second sensor would capture a color of the object at the time of scanning. The output of the scanner was represented in a cloud of points, in which each point had coordinates of the X-, Y- and Z-axes and a 24-bit RGB value for color. The Head Scanner and its example results are shown in figure **6**. [21.]

Figure 6. Cyberware Head Scanner and its results. Copied from cyberware.com [21].

In the 1990's, the company made a step further and developed a successor to the Head Scanner – the Whole Body Scanner. It allowed capturing the full human body shape and its color in only 20 seconds, starting moving down from the head. The Whole Body Scanner and its example results are shown in figure **7**. [21.]

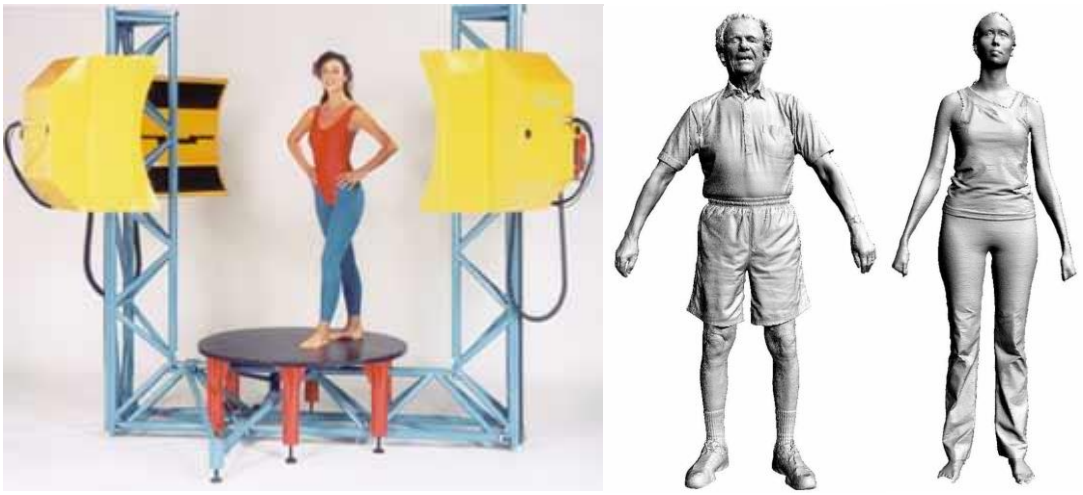


Figure 7. Cyberware Whole Body Scanner and its results. Reprinted from cyberware.com [21].

The scanning solutions of Cyberware Laboratories were very expensive and affordable only to big companies, with the Head Scanner starting from $63,000 and the Whole Body Scanner starting from $200,000 in domestic American sales [22]. Both the Head and the Whole Body Scanners were full-scale working stations due to their sizes and weights. In packed condition, the Head Scanner took 1.9 x 2.0 meters and weighed 127 kg; the Whole Body Scanner was 2.0 x 1.2 meters and 450 kg [23]. They were not easily transportable and demanded setup and calibration after the haulage.

In late the 1990's, many companies, including the earlier mentioned Cyberware, made efforts to expand and develop scanning technologies. Digibotics launched a laser point based scanner that worked in four axes with six degrees of freedom but was slow and unable to trace a color. 3D Scanners introduced REPLICA, a laser stripe based scanner, which made a serious progress in the industry. The Faro Technologies and Immersion companies tried to reduce production expenses of scanners and released low-cost products that were able to create a complete model but were slow and had accuracy that was lower than usual. Another 3D Scanners product, named ModelMaker, was a combined laser stripe scanner with a manually operated arm. ModelMaker was able to create complex and accurate models and color them, while being a flexible and fast device. Its complete scanning process could take just mere minutes. [20.]

Since the middle of the first decade of the 2000's, transportable 3D scanners – such as first generation models of REVscan [24] - started to appear and penetrate into professions not tied to the one specific workplace. They found their own application in different offsite work, e.g. in work of U.S. federal investigators and anthropologists [25]. The 3D scanning became widespread in medicine, in computer tomography [26], in museums and in archeologic work. It was used for saving the cultural heritage in digital form, like a scanning of Kasubi Tombs in 2009 [27] and a scanning of Thomas Jefferson's Monticello in 2002 [28].

Currently, the 3D scanner market is diverse and extensive. Scanners has undergone the process of minimization and became easily transportable by hands in comparison with early scanning stations. A variety of stationary desktop and handheld scanners is available with affordable cost with different scanning techniques involved [29].

Affordability and accessibility of specific electronic components and the widespread distribution of computer hardware allow many hobbyists to develop their own scanning solutions at home. Ordinarily, they publish their designs and drafts on the Internet free of charge with the full description on what components are needed, where to buy the components and how to assemble the working station. Some of them even put low-cost do-it-yourself kits of their scanners on the market for sale.

FabScan is one of the most notable homemade scanners. Francis Engelmann started the FabScan project as his bachelor's thesis with the production cost of 150 euros. Originally, it was based on a laser, a rotary object plate with a stepper motor and a simple web camera. Engelmann released FabScan as an open-source project so that everyone could use and apply any changes to it. [30.] Mario Lucas used the FabScan project as a basis for his own thesis work and extended it into a Raspberry PI platform, shown in figure **8**. By doing so, he minimized the scanner with usage of a native Raspberry camera and separating FabScan with computer hardware. [31.]

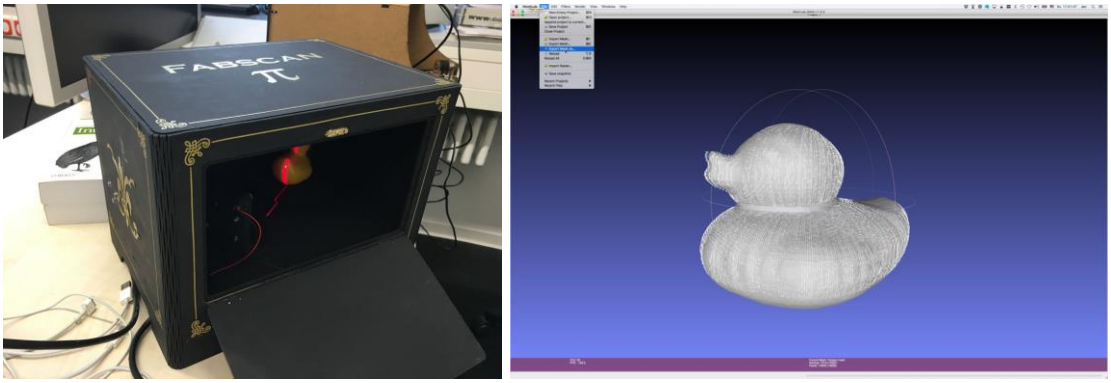


Figure 8. FabScan PI and its result. Copied from Thar [32].

The creation of FabScan has shown that with the certain diligence and determination, a person is able to achieve significant results without access to serious manufacturing capacities and obtain in the result a high-quality product. Due to efforts and ingenuity of Francis Engelmann, FabScan became a great alternative to commercial products on the market. Most importantly, his developments are open to public access and modifying, which will allow this project to live and progress through society contributions.

Homemade scanners are able to compete with featured commercial products even by now and affect the industry by striking to the overall price, thereby forcing companies to release more low-cost products.

Fairly recently smartphone-based scanners were introduced to the market. A rapid evolution of mobile devices and their operational systems allowed them gradually to adopt the functionality of computer systems in some aspects of data processing. Smartphone-based 3D scanners are represented in the majority as a combination of two devices: an independent attachment that does actual scanning and a smartphone, which is used merely as a computing unit. EORA 3D Scanner is an example of combined devices [29]. However, a number of 3D scanners that work only with a standard configuration of

smartphone is drastically small in comparison with the number of combined 3D scanners. 3D scanners aimed for the standard configuration of smartphone are represented as standalone applications, which use principles of photogrammetry. The most notable ones are Autodesk 123D Catch (http://www.123dapp.com/catch), Trnio (http://www.trnio.com/) and Scann3D (http://scann3d.smartmobilevision.com/).

In spite of being of the same type, the way they work differs sharply. All computations for model reconstruction in 123D Catch and Trnio are performed on the cloud [33]. This structure of the application demands a network connection and the ownership of a user account in the service of providing company. In addition, the structure restricts the role of the application only to the image capture and a preview of a completed model. On the contrary, Scann3D executes all computation exclusively on the device, albeit it still allows publishing the model on the Internet [33]. Undoubtedly, the cloud computing benefits in the speed of work, but the 3D scanner becomes divided into client and server parts, which hampers the application support.

The smartphone-based segment of the 3D scanning market is developing slowly. At the time of writing this thesis only three given applications displaying reasonable results were available. Furthermore, only Autodesk 123D Catch was available on all popular mobile operational systems, when Scann3D was limited to Android and Trnio to iOS only [33].

3D scanners have been defined as a convenient tool in various areas of human activity. The American Federal Bureau of Investigation (FBI) uses them for cast creation of crime or accident scenes [34]. Museums and historians use scanners for saving ancient artifacts, relics and remains as digital models [35]. In medicine, they are used for creation of prostheses, dental implants and orthoses. Plenty of professions are able to find a practical application for 3D scanning.

3 Theory of 3D modelling

3.1 3D modelling

A three-dimensional model is a recreation of an abstract or physical object, made in form of a point cloud or a polygonal mesh with connection by diverse geometric entities.

Visualizations of point cloud and mesh are shown in figure **9**.

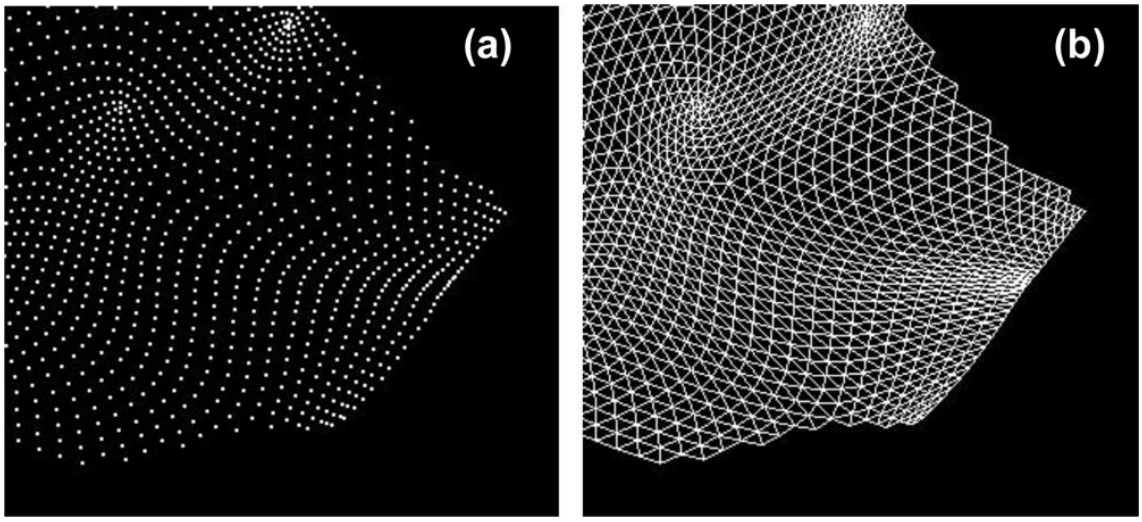


Figure 9. Point cloud (a) and corresponding polygonal mesh (b). Reprinted from Hosoi et al [36].

Point cloud is the simplest representation made of multitude of separate points in 3D space. Typically, point clouds are not used in the modelling but are products of a 3D scanner work which will be converted into a mesh later via different processing algorithms.

Mesh, or polygonal mesh, is a set of vertices, edges and faces that represents the object surface, where a vertex is simply a point in space, where an edge is a line connection between two vertices and a face is a set of closed edges, which can be a triangle face consisting of three edges or a quad face made of four edges. Visualization of the given elements is shown in figure **10**. [37.]

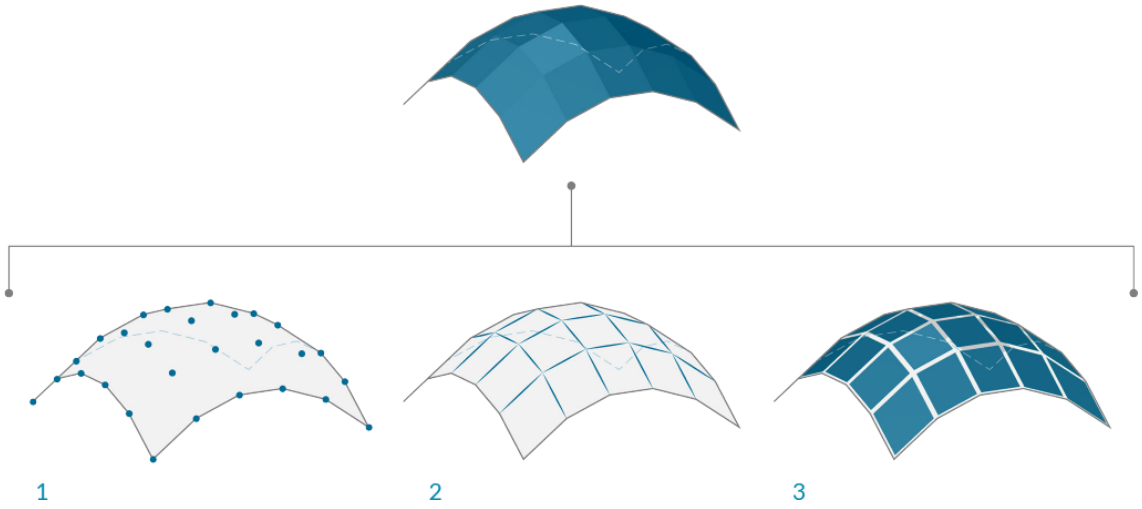


Figure 10. Visualization of vertices (1), edges (2) and quad faces (3). Copied from What is a Mesh? [37].

Usually, models are classified into two categories – solid models and surface models. A solid model represents an object and its physical characteristics, used for engineering and scientific simulations such as load analysis. A surface model is a representation of an object shell without any physical characteristics, and it is used for visualization only, such as in movies and games. [38.]

A 3D modelling is a process of a model creation. A model is visualized as a two-dimensional image via the 3D rendering process or other visualization technologies. There are several commonly used methods for modelling [39]:

- Polygonal modelling, represented by vertices connected with edges for forming of a polygonal mesh. Due to the planar nature of polygons, curved surfaces are approximately simulated by using a multitude of polygons. The polygonal modelling is a popular modelling process because of flexibility, overall lightness and possibility of a polygonal mesh to render in easy and quick way. [39.]
- Curve modelling, where curves are used for the generation of a surface geometry with mathematical equations and are influenced by weight control points. For description of surface forms, the curve modelling may rely on non-uniform rational B-splines (NURBS), geometric primitives and common splines. [39.]
- Digital sculpting, where a model is used as clay during usual sculpting to form an object surface with application of different displacements [39].
- Code-driven modelling, where the code or a procedure generates a model in conditions set by user [39].

- 3D scanning, where a model is generated based on provided data. Different techniques are described in the following chapter.

3.2 3D scanning

A 3D scanning is a process of a capturing object shape with usage of specific equipment and producing a 3D model as a result. The produced model may be in a different form, such as a point cloud, a polygonal mesh, either texturized or colorized.

3D scanners can be categorized by:

- Range of use – short, mid and long ranges, which depends on the manufacturer choice. In general, all stationary desktop scanners with a rotating stand may be considered as short range, because their range of scanning are limited to their physical size.
- Type of an object region scanning – a point, a line or an area scanning, shown in figure **5** on page 10.
- Type of an applied scanning technology.

Scanning technologies are usually divided into two general categories:

1. Contact, where a scanning device is using a mechanical touch probe to determine physical geometrical characteristics of a specific object. Contact technologies were first applied technologies in the 3D scanning and are still used today in modern coordinate-measuring machines in combination with other scanning methods. [40.]
2. Noncontact, which involves a variety of other proximity scanning methods, such as a laser triangulation and pulse, a structured and modulated light, and photogrammetry. The presented noncontact methods include active methods, which emit any kind of a light or a laser signal onto an object, and passive, which do not emit anything and rely on reflected ambient radiance. [40.]

Laser triangulation

A laser triangulation scanner uses two general devices, a laser-generating module and a reading sensor or a camera, for obtaining the distance to the scanning point of reference. Distance to the object point (***Z*** mark in figure **11**) is calculated by applying the triangulation principle with known baseline ***b*** and angle ***α*** between the laser module and

the camera. The angle and baseline parameters determine the aggregate performance of the scanner in aspects of sensing range, occlusion of either laser light or camera's reading due to obstacles or other object surfaces and depth resolution. The laser triangulation technology is one of the most used technologies in handheld and portable devices. [41; 42.]

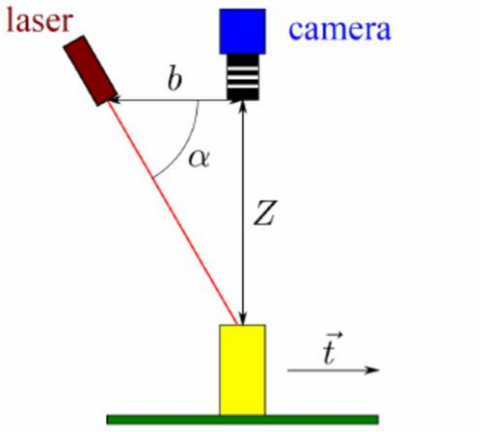


Figure 11. Laser triangulation scanner apparatus. Reprinted from Munaro et al [41].

Laser pulse scanners are based on the time-of-flight principle and are almost similar to the laser triangulation scanners on module level, with application of a laser source and a reading sensor. The accuracy of a scanner depends on the precision of a time measurement between the emitting and receiving laser, reflected from an object surface. Distance to a scanned point is determined by half of the measured time, due to a laser taken a round-trip and a well-known speed of light. [42.]

Structured light

The structured light technology is based on the same triangulation principle but instead of a laser-emitting module, a light source is used. It projects a multiple linear pattern patch to the object and examines a line edge deformation to calculate the distance and form. Projected patterns provide a highly precise depth perception and accurate measurements as a result. The structured light technology is considered as highly accurate and low noise but habitually not very portable. The technology is also sensitive to ambient lighting and limited to an area scanning only. An example of the scanning process of the structured light is shown in figure **12**. [42.]

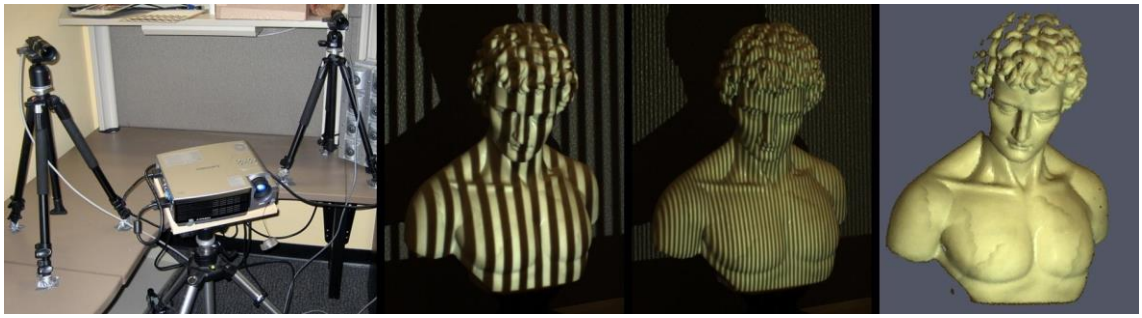


Figure 12. Structured light scanner usage. Copied from McDonald [43].

Modulated light

Modulated light scanning systems use a time-of-flight principle, by analogy to laser pulse systems. The source radiates a light signal with a continually varying amplitude in form of a sine wave. The camera receives the reflected light and determines the distance by analyzing how much the light phase was shifting. In contrary to the structured light, modulated systems are not interfered by the light of other than scanners sources. The technology has the same pros and cons as the structured light. [40.]

A reflectance and transparency of an object surface significantly affect the accuracy of the aforementioned light and laser scanning technologies. The laser signal will not return to the reading sensor after being absorbed with a low reflectance level surface or a permeating translucent surface. [41.] There is no simple and viable solution to this problem but mere workarounds, such as heating the object and acquiring its geometrical shape via a thermal imaging device [44] or coating a surface with a temporary non-transparent spray [45]. An example of a dark-colored object with low reflectance problem is shown in figure **13**.

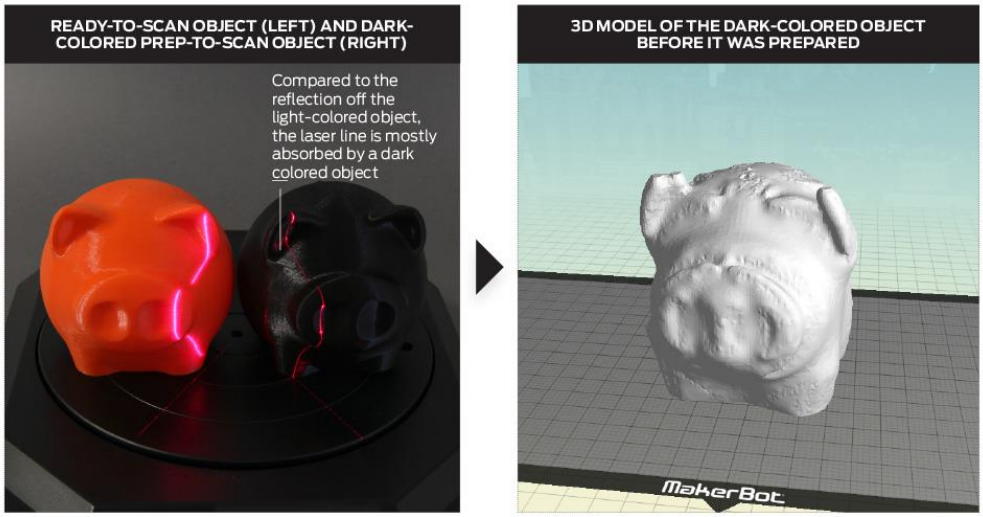


Figure 13. Reflectance problem example. Reprinted from Wohl [46].

Photogrammetry

The photogrammetry is a science of making accurate measurements on the basis of taken photos. It is not dedicated to the three-dimensional scanning only, but involves it as one of the parts of application. A photogrammetry-based scanning relies only on a reflected ambient luminesce and is considered as a passive method. [47.] Due to a vast development of the image processing, it includes a multitude of methods that are used in the scanning [40].

Stereoscopy

the stereoscopy involves usage of two cameras, pointing at the same object and taking pictures at the same moment of time. Slight differences between pictures and a known angle between cameras allow computing a disparity map and a depth map, which are used for determining the distance to an object surface vertices, with a one vertex per map pixel. [40.]

Photometry

The photometric scanning implies taking multiple photos of an object under different specific light conditions, such as with a light source above, below, in front of and behind the object. During the process the object should stay stationary. [40.] Typically, the technology demands setting up a special photometry booth with a great number of cameras pointing under a purposely-calculated angle and a plenty of light sources for obtaining

the best results, which is quite an expensive technology to be built up from scratch, an example of which is shown in figure **14** [48]. Notwithstanding this, low-cost versions also exist with appropriately reduced quality, using only one DSLR camera and four light sources attached to it [49].

Figure 14. Photometric scanning process photoset. Copid from Next Generation Photometric Scanning [48].

Silhouette techniques

The silhouette technique scanning derives object shape in form of an outline from photos, usually taken on high contrast background or chromakey. A surface recreation is achieved by extruding the outlines and finding the intersections between them. Due to this algorithm, cavities and small details, hidden in between convexities, are not detected. [40.]

4 Smartphone 3D scanner prototyping

4.1 Usage of mobile device and sensors

According to statistics of the year 2015, almost one third of the world's population owns at least one smartphone, making it one of the widest spread technologies and the percent of penetration keeps growing, following the increasing affordability of smartphones [50].

Performance of smartphones is rising annually, ensuing the same Moore's law, as the computer system performance [51]. Even now, it is possible to face a plethora of 3D games with a qualitative graphics and a various CAD software, needing considerable capacities during the work that was released specifically for the mobile phones.

The Google Android is a mobile operation system, currently dominating the market with the coverage of 86% of all smartphones in the world. Another popular smartphone operational system is the Apple iOS with coverage only in 12% of the market. [52.] In general, both operation systems are equal in terms of convenience and functionality to an end user. In this situation, it will be better to aim on the Google Android platform during the development of the prototyping application, because the Android platform is more accessible in concepts of cost and it is more widespread, giving an opportunity to test the application on a larger number of devices.

The latest version of the Google Android, 7.1 Nougat, was released on 4th October 2016. In obedience to official statistics of 9th January 2017, it is used only on 0.2% of all Android devices, and the most popular versions are still 6.0 Marshmallow with 29.6%, 5.0 Lollipop with 33.4% and 4.4 KitKat with 22.6% of all Android devices [53]. For achieving the maximum coverage of Android users, the aim should be set to the lowest given version, since all consequent versions have backward compatibility with previous ones. In total, with development aimed to the 4.4 version, prototyped application will be accessible to 86% of all Android users.

A standard configuration of modern smartphones includes front and read cameras with flash, an accelerometer, a gyroscope, a magnetometer and, in some cases, more extraordinary sensors such as a barometer, a thermometer and a dosimeter [54]. The Google Android does not have explicit requirements to specifications of accuracy and

precision of supplied device sensors; thus, hardware implementation varies between models and manufacturers. It is logical to expect that high-priced devices have installed more expensive and accurate sensors while low-cost models have cheap ones with bigger fallibilities. Furthermore, a sensor is a measuring instrument that must be calibrated over the time. Some of the supplied sensors were calibrated during a factory manufacturing and some were not. Regrettably, there are no statistics on this subject. This leads to the fact that the result of the prototyped application work will vary as well between the smartphones. Nonetheless, sensors can be calibrated manually [55] with the use of a third-party application and they can provide quite good results of gathered data.

In spite of marketing specialists determining the camera quality by the amount of acquired megapixels, the amount of megapixels is not related to the camera quality. The megapixel is one million of pixels that forms the image [56]. A photo quality relies on camera optics, a camera sensor and a subsequent processing. Smartphone camera optics are not comparable with professional optics by virtue of their own dimensions and architecture for the sake of mobility and minimization, yet it is enough to produce images of acceptable quality. Indeed, there are certain types of phones with advanced cameras, so called camera phones, but they are very rare on the smartphone market. Optics improvement is an expensive and a sophisticated objective; thus, manufacturers emphasize increasing the number of megapixels [57], not only amongst smartphone cameras but also amongst any other ones. An important factor heavily affecting a photo quality on a mobile device is an autofocus, which is realized by a contrast detection on the majority of smartphones [58].

4.2 Initial composition

In consequence of the fact that the prototyped application was limited by basic specifications and a standard configuration of smartphones, preference to the silhouette scanning technique was given, needing not any additional devices.

The choice of this technique applied further restrictions to the application during the photo shooting:

- object has to be on focus to avoid an unclear silhouette's contour
- object has to be centered in the photo for a truthful juxtaposition of silhouette and a device rotational angle
- object has to fully fit in the photo

- object has to be distinguished on the background, superfluous details must not be recognized as a part of an object
- object and its relief should not cast deep black shadows to avoid their influence on silhouette detection
- photo shooting has to be accomplished in a short time, because a device will not be in totally static condition

In the development of an application roadmap, the first appearing question was to how to take a photo in the best way. How to achieve the best focus of the object and its contrast? The best decision is to delegate this function to an external application, set as default in the user's smartphone, because creation of a meaningful professional photo application is not an objective of this study and deserves a separate and detailed discussion.

During the time of a photo shooting, device rotational angles have to be acquired and stored. Acquisition has to be done on the basis on a smartphone's magnetometer, accelerometer and gyroscope data. Rotational angles may be stored in for example INI-format for a future reference in a model reconstruction. Notwithstanding that the INI-format is considered as deprecated by Microsoft, it is easily readable and modifiable by user in its raw form. Regrettably, Android does not support the INI file format natively and an additional external code library ini4j has to be used for it [59].

Presumably, a taken photo has to be scaled down to speed up a processing calculation on the small powers of a mobile device.

The object of scanning has to be extracted from a background, which means relying on automatic segmentation is risky due to capturing superfluous details or discarding the meaningful ones. The user should be able to mark manually all necessary areas to be discarded or added.

Considering the fact that the camera will move around the object, size of the object will be fluctuating between photos depending on camera approaching and drifting apart. Extracted objects from different photos have to be aligned to the same size for a successful juxtaposition of silhouettes. Due to the unknown distance to the object, a question appeared on how to determine a scaling coefficient of images.

The contours of silhouettes have to be retrieved from objects of the same size. Contour must be a precise and distinct line of one color with the width of one pixel. This format allows transfiguring the contour easily to a point array in three-dimensional space.

Contours, transfigured to point array, should be revolted around axes in 3D space. It is important that contour points were centered regarding the starting points of all axes in space for a successful detection of intersections in the future.

The most logical solution for a search of intersections will be an extrusion of contours. Extrusion has to be accomplished by casting a line through each point of array, which is parallel to the camera view vector with line's center situated directly in the point. The length of the casted line should be more than a maximum distance between the most remote points of the contour. It is a good idea to use the width or the height of a taken photo, depending on what the biggest value is.

Model reconstruction has to be done by a search of intersections between lines of extruded contours. During the intersection of two lines, closest points of lines have to be translated to an intersection point, thereby forming the surface of an object eventually.

The Google Android platform has multiple open source libraries for image processing, but only BoofCV (http://boofcv.org/) and OpenCV (http://opencv.org/) are applicable for all requirements of prototyped application. The OpenCV library was selected because of two factors: it is better in comparison with calculation speed of BoofCV [60] and many large IT corporations, including Intel, Google and Microsoft [61], use the OpenCV. The latest version of OpenCV for the Android platform is 3.1.0 and it limited the lowest usable version of Android to 5.0 [53], decreasing by this coverage to 64% of all Android users.

4.3 Device rotation determination

The Android operation system does not provide direct access over sensors but maintains access through the specific framework. The operation system offers multiple ways of obtaining and handling the sensors' data. Device rotation determination methods typically use a combination of accelerometer, magnetometer and gyroscope sensors. In particular, the Rotation Vector Sensor interface uses all of the mentioned hardware sensors, whereas the Game Rotation Vector Sensor uses only the gyroscope and accelerometer. Thus, the Game Rotation Vector Sensor points to another reference point instead of

binding to the North Pole, and the Geomagnetic Rotation Vector Sensor avoids gyroscope usage, superseding it with the magnetometer. Apart from the given options, the framework allows creating a custom implementation for rotation angles obtainment with utilization of only the gyroscope or any other sensor combinations. The aforementioned realizations provide raw data from sensors that can be processed into a rotation matrix and Euler angles by supplementary framework functions. [62.]

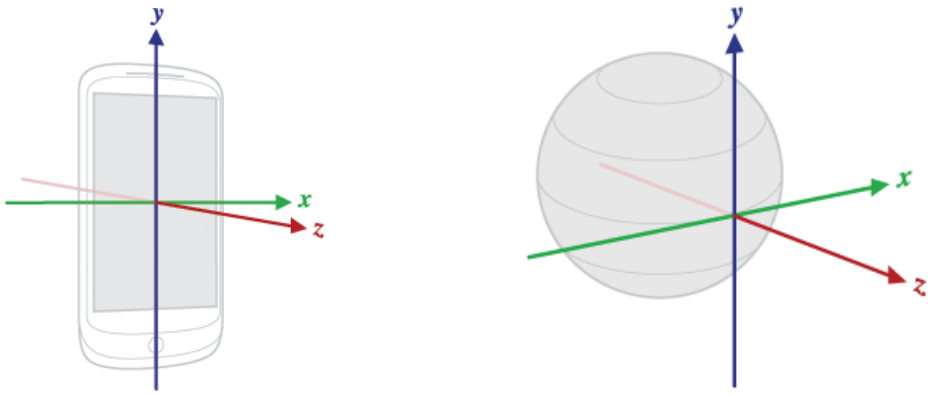


Figure 15. Android device coordinate system (left) and the world's frame reference (right). Reprinted from Android Developers [63].

Android distinguishes two types of coordinate systems. On the one hand, devices use a standard 3-axes coordinate system relative to their display. It is based on a natural orientation of the device, typically portrait for smartphones and landscape for tablets. The coordinate system for the device, which put into its natural orientation, has a horizontal X-axis pointing to the right side, a vertical Y-axis pointing upwards and a Z-axis points towards the outside of the screen, as shown on the left in figure **15**. On the other hand, all rotation determination methods that have a magnetometer involved use a coordinate system relative to the world's frame as a reference, shown on the right in figure **15**. Its Z-axis points to the sky, the Y-axis points to magnetic north of the Earth and the X-axis points roughly to the East. Withal, sensor system axes are not rearranged when device is revolted and this should be done manually if a certain need appears. Rotation angles are calculated based on mentioned axes: pitch as rotation around the X-axis, roll around Y-axis and azimuth instead of yaw around negative Z-axis in Euler form. [62; 63; 64.]

The diversity of sensors also presents several problems coming with them. The gyroscope has a drift around the Z-axis, yet the accuracy of the relative rotation is higher than with the magnetic field sensor. The magnetometer has a magnetic north as a constant point of reference and reduced battery usage in comparison to other sensors but it is influenced by changes in the magnetic field around the device. A heavy usage of multiple

sensors will consume battery's charge at a fast rate, depending on how fast the application queries the data from them [62]. Most angle retrieval methods may have a gimbal lock, loss of one dimension in space, in certain positions [65]. In some cases, it is possible to avoid gimbal lock by rotating the retrieved rotation matrix into a different coordinate system with a designated framework function.

The prototyped application was aimed to use the Rotation Vector Sensor interface for retrieving the device rotation with both good precision and reference to the world's frame that will allow transferring project files between devices and populating one project by images taken from multiple devices while having the rotation angles presented in the same coordinate system. The framework suggests using the device's Z-axis as a sensor system's Y-axis for augmented reality and other camera based applications [66]. By relying on these settings, the smartphone will provide stable and usable rotation angles around the world's frame Z-axis with the starting point on the magnetic north, around the horizontal plane and around the camera view vector or the yaw in any orientation.

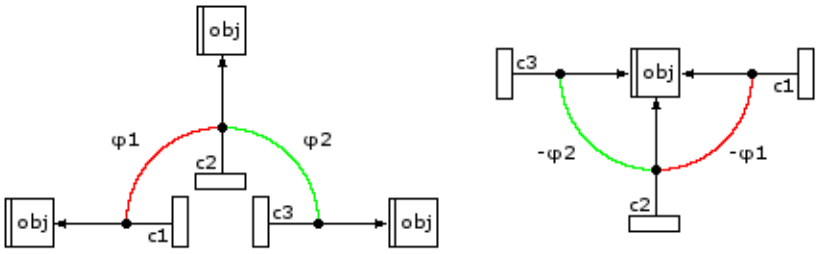


Figure 16. Camera rotation in reference to object, made with free graphic editor GIMP 2 [67].

Even though the retrieved Euler angles may be directly assigned to an object and represent its rotation, they have to be inverted by sign at first. The camera view vector rotates around its own axes during the photo shooting and assigning angles to other objects is not possible in an ordinary case. However, by assuming that a static object will always be at the other end of the camera view vector, angles may be assigned and reverted. Reversion of angles has to be done due to a change of a rotation direction, as if it was not the camera revolting around an immobile object but the object revolted and the camera stand still as shown in figure **16**. Thus, the prototyped application will be in a way similar to desktop scanners that have a rotational stand.

4.4 Man-to-machine interface

The graphical user interface of the prototyped application was developed in accordance to a rule of minimal interaction between the user and application. The user must have access only to the most necessary operating levers of the application and everything else must be hidden under the hood, prepared beforehand or computed automatically. The interface of the prototyped application is limited by the dimensions and resolution of the smartphone display. It is extremely impractical to force the user to adapt and orientate in a complex architecture and intently look into a multitude of small elements. Therefore, an Occam's razor has to be used for cutting out, simplifying and unifying all unnecessary and protruding elements. Elements that remain have to be brought to the intuitively understandable and as standard as possible pictograms wherever feasible.

Since 2014 and the Android version L, the Google has promoted the usage of material design on all devices including the versions where the design has not been implemented originally [68]. The main idea of the design is that application pages open, switch and minimize as "cards", almost invisibly and smoothly with application of shadow effects. Pages avoid sharp edges and angles while being more intelligent and responsive [69]. In the design development, a standard Android material design guidelines were followed.

Basing on the given roadmap of the application, a page with a list of created projects, an interactive page to work with object segmentation and a page to show the result of scanning process were needed as minimum.

The list of projects is shown on the starting page of the application. The list was realized as a standard scrollable list of containers, sorted by modification date, where latest date is always on the top. The container itself consists of the project name, optionally of the date of a creation or a modification and a controlling button that shows the menu with actions. The menu has standard options like a renaming or a deletion of the project, viewing the result and re-doing the segmentation and calculations of the result. Besides the list of projects, the page has a floating action button, responsible for the creation of a new project. A mockup of the project page is shown on the left in figure **17**. All following mockups were made in the free graphical editor GIMP 2 [67].

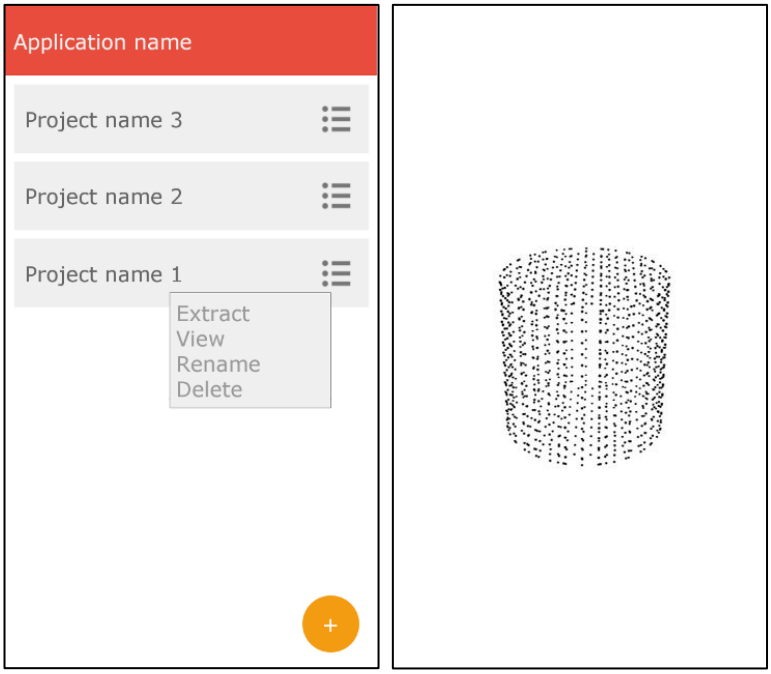


Figure 17. Project page (left) and model page (right) mockups, made with free graphic editor GIMP 2 [67].

The 3D reconstructed model page is fairly simple. A 3D model in form of a point cloud is placed in the center of the page with a possibility to rotate it in all axes by standard swiping gestures. The page does not have any control elements and thus its work and interaction may appear as unobvious for a new user. The new user is introduced with a short description of actions shown in the popup message, with “Swipe to rotate” and “Pinch to zoom”. The page mockup is shown on the right in figure **17**.

The segmentation page architecture is composed of several elements – an original photo of the object, a result of segmentation, a result of contour detection, a drawing area for marking parts of the object and a button panel.

Initially, the photo and resulting images were planned to be shown in the form of a sequential list, but in this case the user had to determine in a very inconvenient way how well contour was detected and if there were any missing or superfluous elements by continually scrolling up and down. For avoidance of a hustle, all elements were placed superimposed on top of each other. The original photo was put on the lowest level, the segmentation result is placed on top of it with opacity of 70%. On the following layer, a result of contour detection was put with the same level of opacity. The subsequent layer is intended for marking elements of the image that should be removed or added. For

easy perception of the user, the green color was chosen for adding and red for deletion. Controlling elements, i.e. buttons, are situated on the highest level.

By means of this approach, an extracted object and its contours will be brightly highlighted and readily discernible, while other parts of the image are partially darkened. This allows easily evaluating the quality of detected contours, existence of noise and parasite details.

Android smartphones have a bottom button bar with 'Back', 'Home' and 'Recent' buttons, also known as soft keys, either in physical form or in emulated form in the bottom part of the screen that will take a vital space of the segmentation screen. It is necessary to hide the emulated button bar, maximize the page to full screen mode and put control elements there with an alternative 'Back' button. Material design guidelines suggest realizing control elements in the lower part of the screen as a solid and monolithic bottom navigation bar that also consumes vital screen space and covers everything beneath it. In this particular case, design guidelines were disregarded and control elements were realized as independent and separately standing buttons of circular form for retrieving larger observable space.

Tentatively, the following control elements were needed:

- a "Back" button for returning to project list page
- a "Drawing tool" button to switch between color filling and pencil
- a "Color" button to change a currently selected color of the drawing tool between green and red
- an "Undo" button to cancel the latest action
- a "Next" button to proceed to the next photo
- a "Compute" button for applying the image segmentation on the basis of user input data and contour detection.

Although the material design suggests the menu curtain should be used on the left side of the display, its exploitation seems to be exceedingly unpleasant because the user has to perennially open and close it for accessing each control element. Hence, control elements were realized in the form of buttons in the bottom of the screen for instant access and have intuitively understandable pictograms for space saving.

Search and selection of intuitively understandable pictograms for a common user have turned into a small study, in the course of which a realization has came that one image can be perceived in completely different ways between people. Due to this fact, only popular pictograms were used, perceiving of which does not cause numerous interpretations and discrepancies. All the chosen pictograms are shown in figure **18**, given on page 32.

Pictogram ‘Menu’, shown in figure 18 (a), is a variation on the theme of famous and ubiquitous ‘hamburger’ icon [70]. A clear indication of the menu, in particular navigation menu, gained a foothold on the icon. The ‘hamburger’ icon was used on the project page for management of a selected project.

Check mark, shown in figure 18 (b), is a conventional concept of consent and acceptance and always used for affirmative or approval buttons [71].

The ‘Undo last action’ pictogram, shown in figure **18** (c), is a classic and longstanding icon in the field of information technology that appeared with the meaning of cancelling the last action since the first versions of Microsoft Word in the 1990’s [72].

The first difficulties appeared in selecting the pictogram for ‘Back’ button. The most popular icon is an arrow pointing to the left and it would be logical to choose exactly it, but by its composition and movement direction it reminds of the ‘Undo’ pictogram. Besides, the ‘Back’ and ‘Undo’ buttons were placed next to each other and the user may become confused and make wrong actions. Hence, for greater differences, an ‘Exit’ icon that represents the door was chosen, which has no discrepancies. It is in use by Google and reminds of a standardized sign of fire exits [73]. It is shown in figure **18** (d).

The ‘Drawing tool’ button had to be represented by two pictograms at once: a pen, for a small detail selection, and a filling for an area selection. Action that has to be performed for direct filling must be enclosed in the filling area pictogram for its clarity to the user. In majority of graphic editing software, the filling area action is realized by one click inside some enclosed form that has to be filled in. The creation of the closed form for the filling execution is excessive in context of the project and it may be done by going from an opposite side and making an inverted filling – to select the object into a rectangular frame with a swipe gesture and fill everything except that. A squared frame icon [74] was taken as the starting point and two points were added, located diagonally and symbolizing a

necessary action to make the filling – to touch the screen and stretch the frame for selection. The produced icon is shown in figure **18** (f).

The ‘Color’ button also demanded two pictograms that fortunately did not present any difficulties during creation. Each of them is an icon of a squared frame [74] filled with either green or red color. The icon is shown in figure **18** (g).

The pictogram for the ‘Compute’ button was the hardest to determine. There is no single established or longstanding pictogram for terms like ‘computation’, ‘process’ or ‘processing’ at this moment. A search engine helpfully suggested the following options:

- a multitude of gears, which are usually associated with ‘Settings’
- a microchip and server stack, unknown to common users
- two, three or four arrows inscribed in a circle that looks similar to the ‘Undo’ button or recycling symbol
- a timer that probably suggests to click the button and simply wait for something.

Uncertainty during the search and a huge amount of options brought an idea of associating computations with a measuring device, shown in figure **18** (h). The scale with an arrow is an essential attribute of every science apparatus and, as analogy, a computation.

(a) Menu [75] (b) Check mark [76] (c) Undo, repeat [77] (d) Exit [78]

(e) Edit, pen [79] (f) Square, fill (g) Color [74] (h) Gauge, meter [80]

Figure 18. Pictograms for application buttons.

Button placement was built in accordance to material design guidelines suggestions about non-standard forms and pages with comparably simple content. It was recommended in the guidelines that control elements should be placed on the right side of the dialog, with an affirmative element situated on the right side of a dismissive element. Other buttons, placed in between of the affirmative and dismissive buttons were subject

to the same logic: the ‘Compute’ and ‘Drawing tool’ buttons were set on right, ‘Undo’ and ‘Color’ were set on the left.

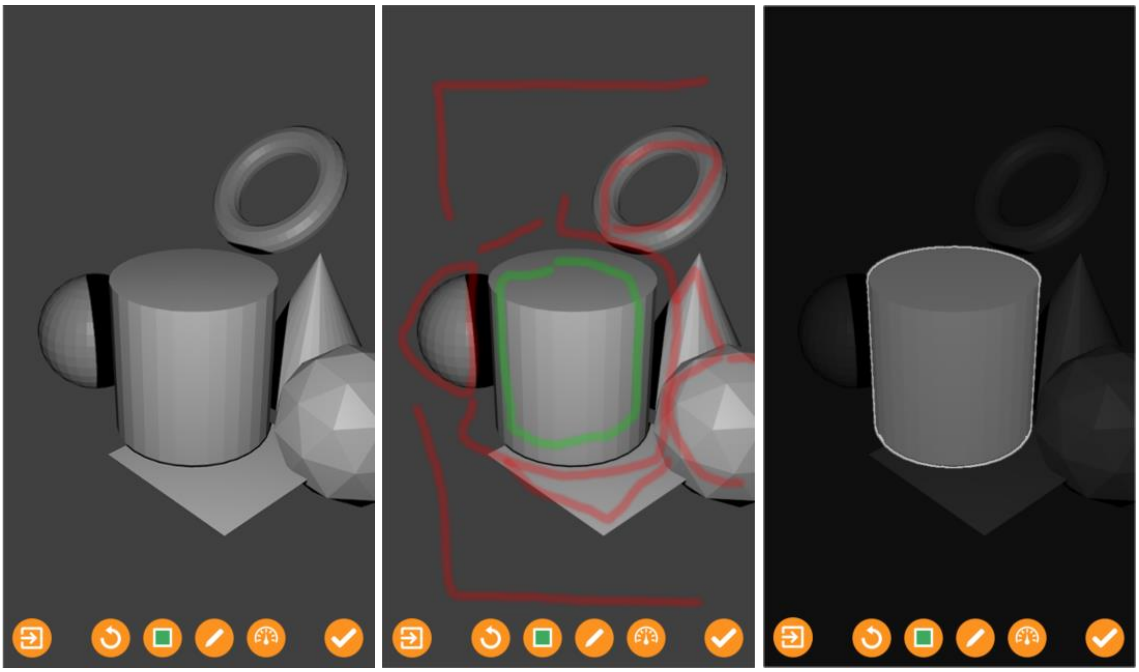


Figure 19. Segmentation page mockup during the process. Original photo (left), user markings (middle), extracted object with edges (right). All images made with free graphic editor GIMP 2 [67].

Interaction with the segmentation page and computed results were visualized with a 3D scene made in the free 3D modelling software Blender 2.78 [81] and the rendered images were processed with the free graphical editor GIMP 2 [67]. The user interface workflow is shown in figure **19**.

5 Image processing with OpenCV

5.1 Object segmentation

Reviewing and selecting the segmentation algorithm was based on understanding that images taken with a smartphone camera could be of mediocre and low quality. Furthermore, there was an expectation that the provided images were done on a complex background and had potential color similarity between them and the object. The segmentation result determines the overall quality of the reconstructed model.

OpenCV provides multiple realized and ready to use algorithms for object segmentation, namely marker-based Watershed and GrabCut, based on Graph Cuts. Both of them have a possibility for accepting markers from the user for achieving a better outcome. Both of them showed good results with nicely cut regions on working with a complex background [82].

Watershed is a segmentation algorithm that treats an image as a topographical map in terms of geography, where brightness of a pixel defines its height and constructs the ridges from them. Resulting basins determine the segmentation of the image. [83.]

The GrabCut segmentation algorithm is based on iterated Graph Cuts. It uses edge and region information encapsulated in a provided picture for the creation of energy functions to retrieve segmentation. [84.]

Watershed separates segmented objects into multiple independent layers, while GrabCut uses two main layers of background and foreground plus one supplementary layer for each. The Watershed ability may lead to heavy over segmentation of the complex picture in automatic mode [83], and the usage of separate layers for each user marking in manual mode. The user has to mark the object with an uninterruptable line to retrieve a single object contour that may lead to problems if the object has a through hollow where the background may be seen, i.e. a toroid figure. Handling multiple layers of the foreground and background demands additional code logic in application.

GrabCut showed better results than the Watershed algorithm during segmentation of object that had color similar to the color of background. Watershed may consider and include into foreground layer parts of the background if color does not have a drastic

difference [82]. The solution to this problem may be using fairly definitive markers. However, even with them the object edge may vary randomly unless markers stand close to each other.

On the other hand, GrabCut segmentation is much slower than Watershed [82], inheriting this issue from the Graph Cut ancestor.

In the end, GrabCut scored in all positions against Watershed, except the speed of calculations. Comparing quality and rapidity, quality won and GrabCut was chosen. It seemed to be more promising and provide better results than the Watershed algorithm.

After implementation of essential code for GrabCut utilization, algorithm performance was inspected on a real device by launching segmentation calculations for a 20-megapixel image without any markers. A Sony Xperia Z1 phone was used for the testing. The total duration of segmentation was approximately 45 minutes, which is unacceptable for application flow. The processing time decreased to 15 seconds when image size was reduced to 1.1 megapixel (1280 per 920 pixels) with user markers and selection of the object. At first, this issue induced to do auto-scaling of all images to 1 megapixel size, but this solution would restrict the usability of the application and ignore constantly growing performances of new smartphones. Therefore, the application lets user the set the image size in his/her photo application and display a warning that suggests that it will take longer to process large size images before shooting.

The OpenCV GrabCut function returns an 8-bit image matrix as a result [85]. It has a filled contour of the object that looks completely black to the human eye. The GrabCut function marks the background with a value of zero, black, and the foreground object with a value of four, also black. The foreground values should be set to true white color, value 255, for simplifying the following processing and human control over it.

5.2 Referenced scaling

Translation of extracted object shapes to the same size appeared to be the most complex task to solve during development of the 3D scanning application. Nevertheless, a working solution to the problem was found.

The size of the object on the first image is taken as a standard one and it is used in all subsequent images. The scaling coefficient for each image is calculated based on the ratio between the heights of the object on the current and previous images. Before the calculation of the scaling coefficient, the difference between rotation angles of camera view vector of the current and the previous images must be computed. The current image must be rotated on the difference to neutralize the smartphone rotation around the camera view vector.

The process of angle calculation is visualized in figure 20, where the object of shooting is shown symbolically as a star [86]. The camera that took image B is shown with a camera symbol [87] and the camera's view vector is shown in form of vector AO. The calculated rotation angle is shown as angle φ.

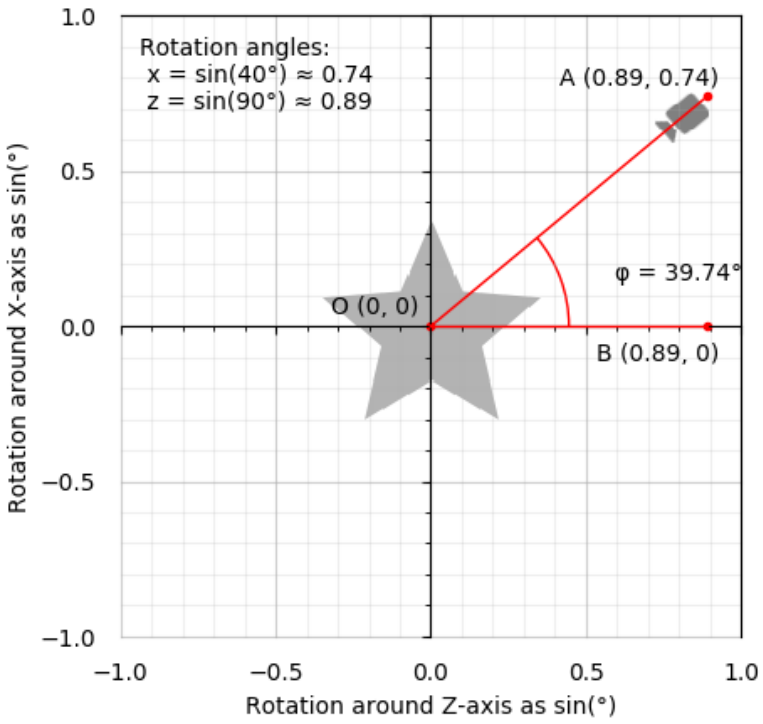


Figure 20. Rotation angle visualization, made with matplotlib [88].

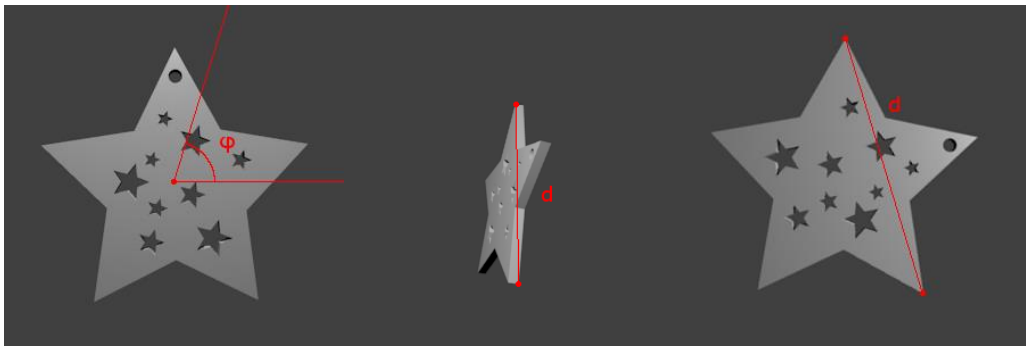
For a better perception of the algorithm a test case with a standard image A and a scalable image B will be described. At first, the distance between the top and bottom extreme points of the object has to be found on image B. In this case, pixels are treated as distance units due to the lack of original physical units.

Image A should be rotated on a specific angle that represents the camera view vector for image B from the point of view of image A. Inasmuch as a smartphone roams in 3D space and operations are done with 2D images, two 3D space angles of the X- and Z-axes have to be brought into one 2D space angle. Rotation around 3D Z-axis is considered as a common horizontal X-axis and a rotation around 3D X-axis as a common vertical Y-axis. This leads to a condition in which each point of the axis has a value of two angles at once, and the starting point is shown as 0°, 180° and 360° accordingly. As a solution, the sinus of an angle is used that has one value for the previously mentioned examples to avoid cluttering.

After bringing the rotational axes into 2D space, a new point, i.e. A, is created, where the sinus of the 3D X-axis angle determines its position on the 2D Y-axis and the sinus of the 3D Z-axis angle – position on the 2D X-axis correspondingly. Two new points are needed for greater convenience, point O in the center of axis and the second point B on the X-axis. At this moment, there are two ways – to convert points into vectors AO and BO and to find the angle between them or apply the law of cosines to three points with the same result. Image A will be rotated on the retrieved angle and extreme points will be obtained.

The ratio between objects' heights is a scaling coefficient of image B. It is also necessary to take into account a perspective influence on the image object, which adds more problems during coefficient calculation. Yet, no solution on how to get rid of that influence has been found.

Thus, this algorithm is applicable to all images in the project, starting from the second one, since the first one is used as a reference of standard size.



(a) (b) (c)

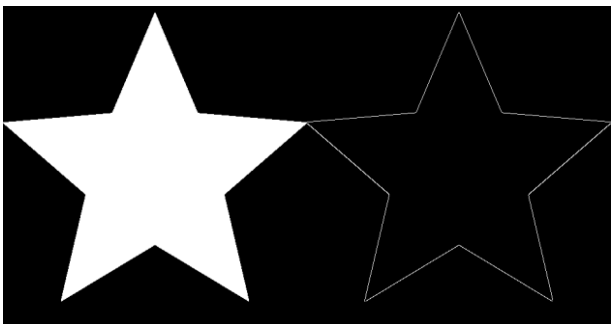
Figure 21. Referenced scaling visualization, made with Blender [81] and GIMP 2 [67].

Another 3D scene was created in Blender [81] for better understanding of algorithm work with two cameras and one free of charge 3D model [89], shown in figure 21. Cameras were located at various distances from the model and rotated under different angles imitating accidental smartphone placement during shooting. Standard image A with the visualized camera vector of image B and the found rotation angle φ are shown in figure 21 (a) and scalable image B with marked extreme points and distance d between them are shown in figure 21 (b). In addition, small perspective influence on the object is noticeable. Figure **21** (c) shows image A, which has already been rotated, with retrieved extreme points and distance d between them.

5.3 Contour recognition

The Canny edge detector is the most suitable for contour recognition on the black and white mask of the segmented object that was retrieved from GrabCut.

The Canny operator, also known as the Canny edge detector, considered as an optimal edge detector due to the low level of inaccuracy, precisely determines the center of the edge and every edge marked only once. It was developed by John Canny in 1986 and still is considered as the best algorithm of its kind. [90.]



(a) (b)

Figure 22. Canny edge detector with both thresholds set at 250.

OpenCV realization of the Canny edge detector allows specifying the value of upper and lower thresholds which are used directly for edge determination. A pixel is defined as edge if its gradient is above the upper threshold and it is discarded if gradient is below than lower one. [90.] Due to the fact that the object mask is 8-bit and that the color value

varies between 0 as true black and 255 as true white, it makes sense to set both thresholds as close as possible to the white color value that represents the object shape such as 250. The usage of the designated threshold shown in figure 22, where (a) is a source object mask and (b) is a Canny edge detector result.

5.4 Transfiguration to point cloud

Due to the fact that computer images are n-dimensional digit arrays, necessary elements may be found through simple iteration. A Mat element of the OpenCV library represents an n-dimensional dense array that is used for storing and modifying loaded images, which operates with convenient table concepts, known as rows and columns [91].

It is important to know which of the X-, Y- or Z-axes are considered as a vertical axis during the conversion of image into points of 3D space. A uniform standard accepted between the formats of 3D models and modelling software simply does not exist. Blender [81], Autodesk 3D Studio Max and Unreal Engine see the Z-axis as the vertical axis, when Unity Engine uses the Y-axis. As it was told previously in chapter 4.3 Device rotation determination, Google Android API uses world's frame Z-axis as the vertical axis, pointing to the sky.

For avoidance of confusion strengthening, the application will keep utilizing the mentioned Android definition of the Z-axis as the vertical axis in the 3D space of the reconstructed model that also will lead to direct usage of the obtained rotation angles without any swapping of axes during transfiguration.

Spatial coordinates of an element can be identified via matrix iteration and via using element's position in the row as the X-axis value and its position in the column as the Z-axis value. The starting point is located in the top left corner of the matrix and the direct usage of the column position will lead to vertical mirroring of the object. A solution to this problem will be a utilization of difference between the total number of rows and the current column position as s coordinate.

6 Point cloud processing

6.1 Adjustment and extrusion

A reconstruction process is erected on searching for intersections between extruded contours. Coordinates that were obtained during the transfiguration process may vary in space in dependence to size and location of the object on the image. Due to this, extruded contours have to be centered by height and width to the origin of coordinates. Otherwise, they will intersect only partly or not at all.

The quest for the method of giving volume to the 2D planes in the 3D space was the second problem to solved in this project. In the light of very limited sources available for model recreation, it was very hard to find a way or technology that may be applied. From non-contact passive technologies listed in 3.2 3D scanning chapter at page 17 only the silhouette technique was suitable to be used in the project. In this way, the scanning application will be somewhat similar to the realization of the 3D scanner of same technique done at the University of Linköping by Karin Olsson and Therese Persson in 2001 [92]. Olsson and Persson used a desktop computer and a rotating stand [92].

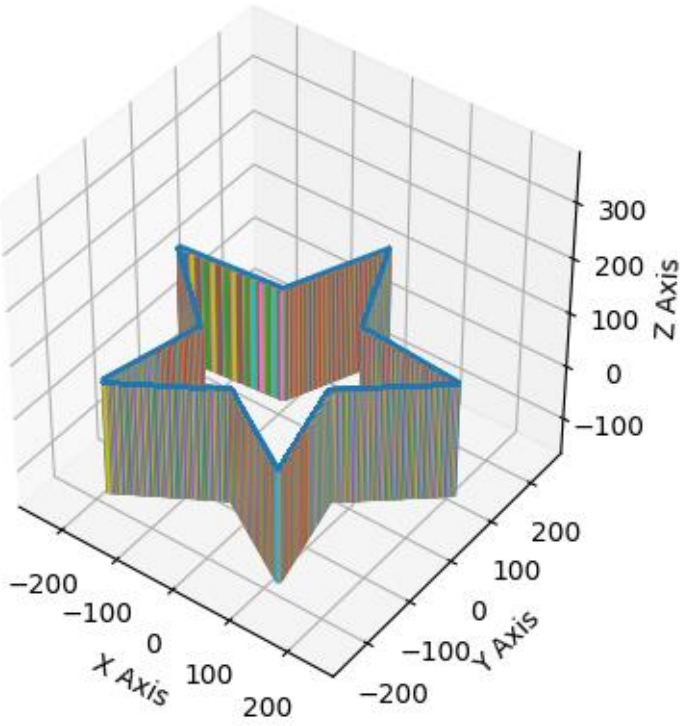


Figure 23. Star contour after extrusion process, made with matplotlib [88].

The extrusion process is represented by the creation of a line segment that passes through one of the detected contour points by determining its location on the Y-axis, inasmuch as the locations of the X- and Z-axes were defined by the point itself. Ascertainment of negative and positive points at the Y-axis is done through utilization of a maximal value of taken photo dimensions, either height or width. Since pixels are acting role of measure units for distance and the object is located in the center of photo, it will allow avoiding the appearance of line segments not reaching each other and as result it will not lead to loss of the intersection point. On the other hand, any six-digit number could be used here. An example of the extrusion process is shown in figure **23**.

6.2 Revolution and recreation

The revolution of the line segment points made during the extrusion process was done with basic mathematics. Undoubtedly, rotation in three-dimensional space is more difficult than in 2D where everything is rotated only around the Z-axis. The rotation axis has to be chosen and rotation is applied to one point twice by the X- and Z-axes with formulae, shown in formulae 1 [93]. Rotation around the Y-axis is not relevant here, as it was already described in chapter 5.2 Referenced scaling.

$$y' = y \times \cos q - z \times \sin q \qquad x' = x \times cos\, q - y \times sin\, q$$
$$z' = y \times \sin q + z \times \cos q \qquad y' = x \times sin\, q + y \times cos\, q$$
$$x' = x \qquad z' = z$$

Formulae 1. Rotation formulae: around X-axis (left) and Z-axis (right). Data gathered from Owen [93].

Model recreation is based on the search of intersections between lines of different extruded contours. In 2D space this search is not difficult to accomplish because all entities are operating on the same level of the Z-axis, but in 3D space everything becomes more complicated than usual.

Ordinarily, lines in 3D space do not intersect at all or have an intersection somewhere in the infinite point of space. In the case when two lines do not intersect at one specific point, they may be interconnected by as short an additional line segment as possible. The set distance threshold of the segment will determine whether the lines intersect or not. At this point, not one but actually two points of intersection will be found located on two provided line segments. [94.]

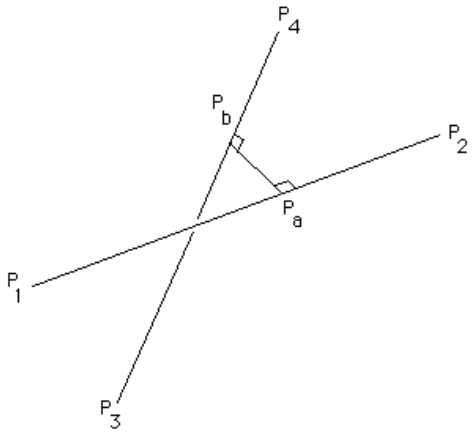


Figure 24. Shortest distance between two lines. Reprinted from Bourke [94].

Figure **24** shows two lines P_1P_2 and P_3P_4 with defined points P_a and P_b that are closest to the intersection. P_a is determined by formula $P_a = P_1 + mu_a(P_2 - P_1)$ and P_b with a similar one $P_b = P_3 + mu_b(P_4 - P_3)$. The mu_a and mu_b values are varying in an infinity range, and the mu values of line segments amid P_1P_2 and P_3P_4 range from 0 to 1. At this point, there are two alternative options on determining the shortest possible line segment amid P_a and P_b points – either to acquire the P_a P_b line segment length and get its minimum or find a line segment that is perpendicular to given lines. [94.] The options allow to write dot products equations as

$$(P_a - P_b)\ dot(P_2 - P_1) = 0 \text{ [94]},$$
$$(P_a - P_b)\ dot(P_4 - P_3) = 0 \text{ [94]}.$$

Extension of dot product equations with equation of lines will look as

$$\left(\mathrm{P_1} - \mathrm{P_3} + mu_a(\mathrm{P_2} - P_1) - \mathrm{mu}_b(\mathrm{P_4} - \mathrm{P_3})\right) dot(P_2 - P_1) = 0 \text{ [94]},$$
$$\left(\mathrm{P_1} - \mathrm{P_3} + mu_a(\mathrm{P_2} - P_1) - \mathrm{mu}_b(\mathrm{P_4} - \mathrm{P_3})\right) dot(P_4 - P_3) = 0 \text{ [94]}.$$

Extended dot product equations may be unfolded in terms of coordinates to the following form

$$d_{1321} + mu_a d_{2121} - mu_b d_{4321} = 0 \text{ [94]},$$
$$d_{1343} + mu_a d_{4321} - mu_b d_{4343} = 0 \text{ [94]}.$$

Where

$$d_{mnop} = (x_m - x_n)(x_o - x_p) + (y_m - y_n)(y_o - y_p) + (z_m - z_n)(z_o - z_p) \text{ [94]}.$$

The mu formulas may be finally produced on the basis of extended dot product equations as

$$mu_a = (d_{1343}d_{4321} - d_{1321}d_{4343}) \div (d_{2121}d_{4343} - d_{4321}d_{4321})$$ [94],

$$mu_b = (d_{1343} - mu_a d_{4321}) \div d_{4343}$$ [94].

The length of the shortest line segment can be calculated on the basis of mu_a and mu_b and compared with set distance threshold for intersection determination. [94.]

The given method is tailored for working with infinite lines and not limited line segments. Direct application of the method will cause appearance of intersection points in places, which line segments should not reach. In order to make sure that an intersection point was determined somewhere close to the intersected line segment, two triangles have to be created and their perimeters compared. The first triangle determines the maximal allowed distance to any future intersection point and is built upon two intersected line segment points and the distance threshold. The second one is built on the determined intersection point and the same line points. The intersection point is considered to be related to the intersected line segment if the intersection triangle perimeter is less or equal to other one.

Lines of extruded contours should belong to independent arrays for avoidance of the intersection search between parallel lines of the same contour. A model billet is made by determining the intersections between the first and second contour arrays and saving the determined ones into an independent array. The model billet has to be formed with the first two images that have an angle difference in the X- or Z-axis of at least 90° to allow the application to cut away as much as possible before starting a further process. Line iteration should start from the third contour array, search for intersections with all previous intersections and later adjust previous contour lines to intersection points. Hence, it should form the model in the same way a sculptor carves a statue from marble. Figure 25 shows a simplified version of reconstruction process casted to 2D space. On the left chart, the second contour array (green lines) searches for intersections with the first contour array (red lines) and finds them, marked as four black points. On the right chart, the contours are mapped to the new coordinates of the intersection points. The third contour array (blue lines) is applied to the first two modified contours and it successfully searches for its own intersections, marked once again as four black points. The process continues until the last of contour arrays.

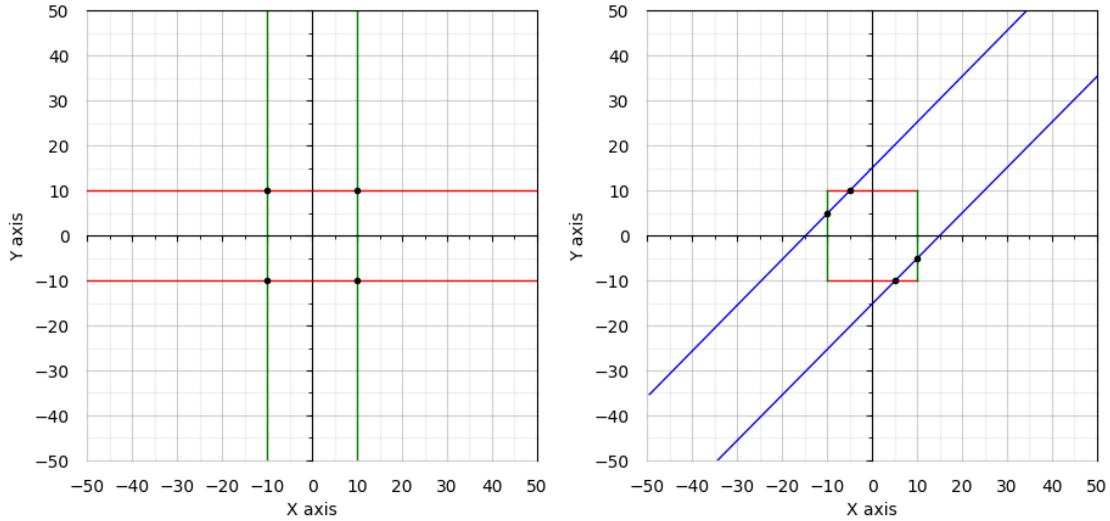


Figure 25. Simplified reconstruction process, made with matplotlib [88].

When the object has complex forms, the line segment will have multiple intersection points. Thus, a bringing of its end points to intersection ones becomes impossible. Since nobody can guarantee that the intersection points will appear in order, the obtained points have to be sorted by increasing the distance from the line segment starting point and later they have to be used for creating sub line segments. In addition, a case must be taken into account, when the contour edge will have multiple consequent points instead of only one point, which may be considered as starting ones. It will ruin the order of the intersection points and the surface created with sub line segments will not be corresponding to real one. Hereupon, intersection points that have a distance of one unit or fewer units between points must be adduced to one point, closest to the starting point of the line segment.

Each point of intersection stores the reference to the line it intersected. After being sorted by the increasing distance from the starting point and forming the actual line segments, old intersected lines have to be adjusted to a new point location. Two intersected lines connected with a new intersecting line may not be connected at a point anywhere else and may be even parallel in one of the spatial planes, however, they are expected to converge at one of the object edges. A problem appears on how to determine which of the intersected lines points has to be adjusted. As it is assumed that two ends of line segments are converged to the object edge, the sum of angles between the points and the intersecting line will be smaller than the sum of others.

The line segments will not reflect the real surface of object after conversion into point cloud because only two points are available and hence a long line segment will be shown as two independent and non-related points in space. The conversion process has to comprise creation of additional points between the starting and ending point of the line segment with the pre-set step. The step value should be selected carefully due to a drastic increase of points and workload on the application and the device.

The amount of taken images influences the smoothness of the reconstructed model. As an example, if the user takes only two images of a round ball, one image in front of the ball and second image from the side of the ball, the application will reconstruct the ball only into the shape of rounded cube due to lack of shape information.

6.3 Adduction to OBJ format

The final goal of the smartphone scanner application was point cloud formation and storing it as OBJ industrial standard. The point cloud retrieved during the recreation process may be adduced to the OBJ format quite easily.

```
# Vertex list

v -0.5 -0.5 0.5
v -0.5 -0.5 -0.5
v -0.5 0.5 -0.5
v -0.5 0.5 0.5
v 0.5 -0.5 0.5
v 0.5 -0.5 -0.5
v 0.5 0.5 -0.5
v 0.5 0.5 0.5

# Point/Line/Face list

usemtl Default
f 4 3 2 1
f 2 6 5 1
f 3 7 6 2
f 8 7 3 4
f 5 8 4 1
f 6 7 8 5

# End of file
```

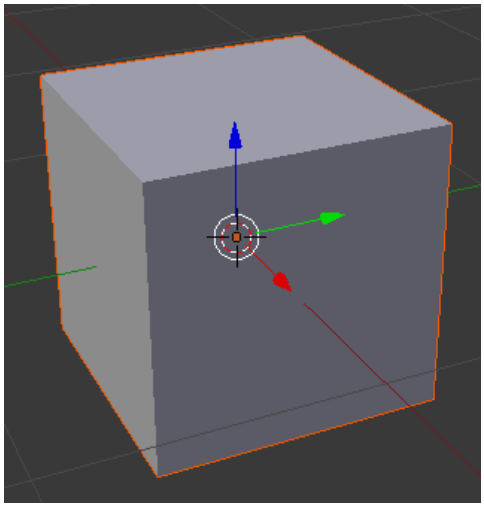


Figure 26. OBJ file example, text version (left) and rendered object (right). Data gathered from Bourke [95].

Initially, the OBJ file standard was developed by the Wavefront company for its own products but later the standard became widespread in the industry. The standard defines the OBJ file format as a plain ASCII text file. The file name must not have spaces which should be changed to other printed characters. It uses all common three dimensional elements, like vertices, line, edges, surfaces and curves and supports both free-form and

polygonal objects. Vertex is determined by the *"v x y z*" line of text, where x, y and z are its coordinates in space respectively. Figure **26** demonstrates the example box in its text and rendered forms. [95.] Point cloud may be stored in this rather simple form for future use in any of 3D modelling software.

7 Implementation results and discussion

The final application turned out to be a complex product with multiple components and, in consequence, all these components have to be tested independently before the first full system check. Testing might be done through unit or integration tests. Hardware data retrieval may be tested only as a real-life measurement. In particular, the following components demand more scrupulous examination:

- Rotation angle retrieval,
- Object segmentation and image processing,
- Referenced scaling,
- Point cloud processing,
- Intersection search algorithm,
- User interface workflow convenience.

Due to severe time constraints, several important modules including the referenced scaling and intersection algorithm have not passed comprehensive testing, leaving a possibility for an exceptional workflow case of any kind to appear at any moment of module work. Moreover, the application has not been tested on as many devices as it was planned initially.

The rotation angle retrieval implementation showed very good results on the Sony Xperia Z1 device with Android version 5.1.1. In a fully static condition, the angle varied within the boundaries of ±0.3°. When the device was held by a human hand in an approximately static position, error rate increased up to ±1° due to jittering of the hand. It is possible to increase the reading rate of the sensor and filter obtained data for compensating the human factor. After changing the position of the device, the rotation angles took some time to stabilize. The angles of the X-axis, horizontal, and the Y-axis, along the camera view vector, stabilized almost momentarily, but the Z-axis took up to 10-15 seconds to get the first stable values. It may become a problem to an end-user, if images are taken in short intervals with assignment of un-stabilized angles and as a result reconstruction of a highly distorted model. The application should notify the user with a signal or popup message that angles are stabilized and an image may be taken.

Furthermore, the used rotation vector interface includes utilization of a magnetometer and its data may be affected by external magnetic fields. Particularly, the data error rate

increases when the device has WI-FI, mobile internet connection or GPS turned on depending on its power. However, even with listed influences from the inside, the present method gave high accuracy of acquired data. Testing was carried out in house conditions and thus it may not be guaranteed that the device was not under the impact of powerful external magnetic fields due to an absence of any equipment that is able to record power of the present magnetic field. The solution to the external influence problem may be measuring of the surrounding magnetic field by means of an embedded magnetometer for its power and variability and switching to magnetometer-free rotation interface in case of high results on any of given parameters. This solution will also lead to inability to extend the project by adding more images or using the project as an extension to other ones.

In addition, testing revealed a gimbal lock position and disorientation in space when rotation angles start fluctuating erratically in a device position close to a flat or reverse flat, precisely after 87° and -87° around the horizontal axis. As it was stated in chapter 4.3 Device rotation determination, this problem may be solved partially by swapping the sensors axes. After swapping the sensor axes, the rotation values stop fluctuating. However, swapping of axes increases rotation values by unknown value, thus producing a gap in comparison to previous rotation values and require more rigorous tuning.

Rotation angle retrieval is done right after returning to the scanning application from the user defined photo application. This implementation creates a certain time gap between image creation in device memory and recording of rotation angles and rather is a prototyping shortcoming than a real problem. Nonetheless, the time gap may be reduced by recording the angles in a background thread into an array sorted by a timestamp and picking the timestamp of the files and appropriate array data upon returning from the photo application.

Object segmentation of the 1.1 megapixel images takes up to 15 seconds and may be considered as an acceptable result. Yet, testing demonstrated that sometimes it misfires and produces small maculae with the size of several pixels, created on the edge of user markers, an example which is shown in figure **27**. Maculae appear in a random place and their amount is random too. This is a serious problem that affects referenced scaling and the following reconstruction process steps. The maculae may be removed manually

by creating additional covering markers but this implies multiple repeats of the segmentation process need to be made. Another possible solution could be searching for small objects by means of OpenCV and following removal of them.

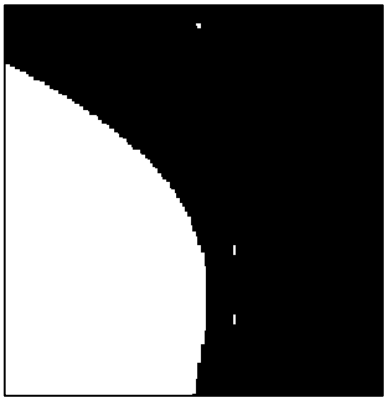


Figure 27. Maculae example.

Object segmentation does not always give the best results by including superfluous details into the object that also leads to the creation of additional user markers and process repeats. On average, each image has two to five segmentation runs, which increases the average processing time of one image to a bit more than one minute.

Referenced scaling was tested in real life conditions with several objects and under different rotation angles. It showed very good results with an error rate of resized image up to ±1 pixel. The referenced scaling process heavily relied on the truthfulness of rotation angles and image masks cleared of any maculae.

Point cloud zero adjustment, extrusion, rotation and intersection search testing did not show any particular problems in unit testing conditions. Simple mathematics always works flawlessly. However, the intersection search algorithm may have a lot of exceptional cases that could appear only in full project testing.

User interface workflow proceeds well for the end-use, yet it has several small structural imperfections. The Floating action button on the Project page hides the control button of the lowest project in the list from the view with impossibility for accessing it if the project list fits the page. Another issue arose from the long time to process object segmentation and model reconstruction, when the page simply freezes and not respond to any commands. A popup message blocking the screen with the current status of the process should be shown to make the user aware of the situation. The segmentation page should

also include a possibility to zoom in and out of the image to check and mark small details. In addition, a small survey was done about the pictogram meanings on the segmentation page. Several people were asked about how they understood the pictograms with a short description of the related process and the survey showed that four out of six pictograms were understood mostly correctly and the others not. The most controversial ones were the Process and Color choose buttons. Some of the respondents defined the Process pictogram as something related to measuring or modifying angles, area of object or whole image.

The first full application check with mockup images of ideal sphere showed problems with adjustment of old intersections. A model billet made with only two images taken with 90° angle between each other around the vertical axis is shown on the left side of figure 28. In the middle of figure 28 a model is shown, which was made with four images taken with 0°, 45°, 90° and 135° angles around the vertical axis without adjustment of old intersections. The same model but with adjusted old intersections is shown on the right of figure 28. Figure **28** was adjusted to be brighter than the original for better perception of the models.

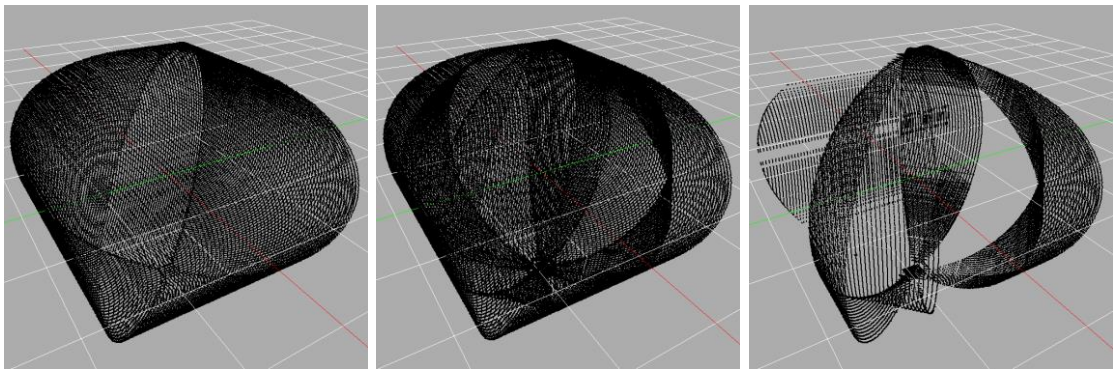


Figure 28. Model billet (right), model billet with found intersections (middle) and model billet adjusted to found intersection (right).

The model billet on the left looks like a rounded cube due to lack of object form information. The model billet in the middle with new found intersections became to resemble spherical shape. Yet, it is clearly seen on the right billet that the adjustment algorithm chooses the wrong points to adjust and this results in a distorted model and disappearance of a multitude of object edges and surfaces. This problem became the stumbling block of the whole project as any further intersection search becomes useless whereas unadjusted lines will invoke an appearance of new intersections in places where they do

not belong. Nonetheless, the model billet made with only two images shows good results with usage of available information about the project.

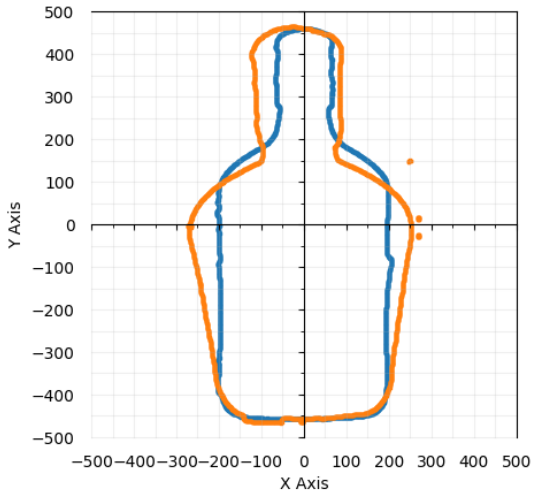


Figure 29. Perspective distortion with maculae inclusion.

Another particular issue that was encountered in the project is perspective influence. Figure **29** shows two scaled contours, a blue colored one taken in front of the object and an orange one with a top down angle, thus having width expanded to the top part of the object and squeezed bottom. In spite of this problem being serious one, it is not as vital to the overall project as wrong adjustment of old intersection and may be easily avoided with keeping the device perpendicularly to the ground. In the given example, the original blue figure will be affected only on the “shoulder” part of the object and nowhere else. Perspective distortion may be corrected by means of OpenCV, yet it has to be computed in some way. Most of the correction formulae rely on the camera lens parameters that differ between lenses and should also include device rotation angles in reference to an object.

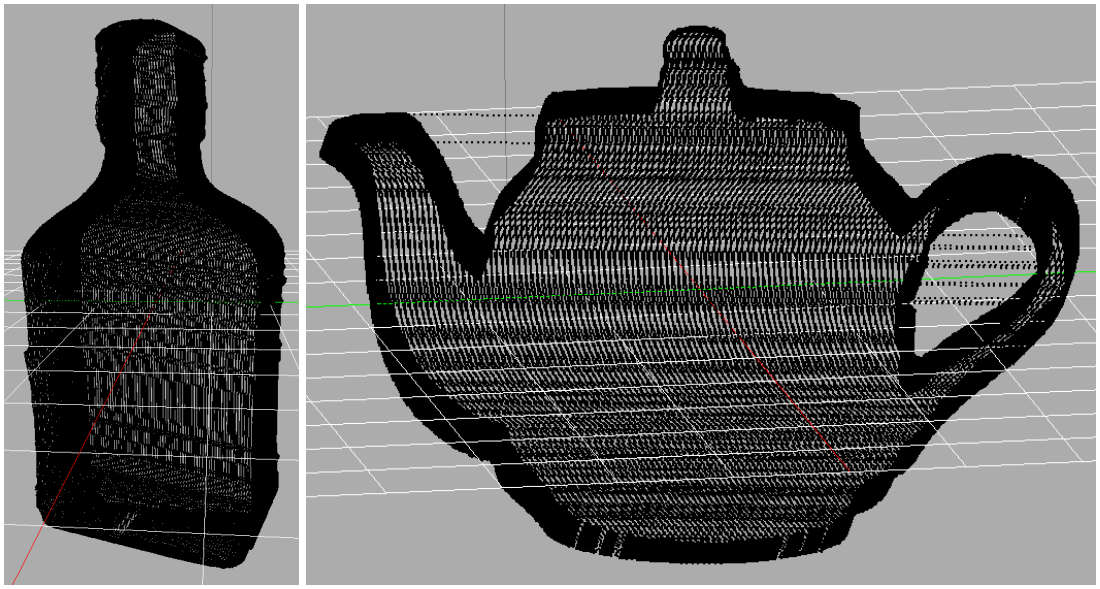


Figure 30. Model billet examples.

Figure **30** demonstrates two examples of model billets made with two images, a bottle on the left in perspective view and a teapot in orthographic view. Superfluous line segments are observed on the teapot figure, multiple of them inside its handle and two passing from its spout to lid. The line segments appear due to a logical imperfection of either intersection search algorithm or the algorithm that adduces multiple consequent points with a short distance and thus breaking the line segment sorting. Figure **30** was also adjusted to be brighter than the original for better perception.

The final application has not met the initial expectations and it only partly met the set goal. It provides good results in the form of a model billet on the basis of only two taken images, as shown in figure **30**, though any further recreation and transformation of the billet into a meaningful model is impossible due to aforementioned problems. The model billet was made on average in 20 seconds and had at least 100,000 vertices of the surface, depending on the size of the images taken. Some of the vertices appear as duplicates in the same coordinates and may be filtered out at the last stage of reconstruction, after conversion to the point cloud to avoid breaking the connecting lines.

8 Conclusion

The aim of this final year project was to develop a fully working mobile phone based 3D scanning application. The application was developed for the Android operational system and smartphones of a standard configuration. The application used the non-contact silhouette technique for the 3D scanning of objects. The silhouette technique uses detected contours of the objects, extrudes the contours and finds the intersections between the contours.

The case study described in this thesis demonstrates that the development of a 3D scanning application that only uses a device photo camera and rotation angles for the model reconstruction is possible and involves multiple techniques based on usage of external libraries and mathematics. The OpenCV library was used for the image processing in the project. The library provided all necessary functions for object segmentation and contour detection.

The objective of the project was met in a semi-successful way. The application allows recreating a 3D model of an object on the basis of taken images, yet problems of perspective influence and old intersection adjustments remain and do not allow reconstructing the model in an accurate way. Due to these problems, this application cannot be compared to any released commercial and non-commercial software. However, it may be used as a starting point for future development and searching for solutions to the aforementioned problems and, at some point, for becoming a sustainable and competitive product. Overall, the project bar was set very high and overall it was challenging, demanding and time consuming.

References

1. Oppenheimer Robin. William Fetter, E.A.T., and 1960s Computer Graphics Collaborations in Seattle [online]. ACADEMIA; 2005.
   URL: http://www.academia.edu/7801224/William_Fetter_E.A.T._and_1960s_Computer_Graphics_Collaborations_in_Seattle
   Accessed 2 December 2016.

2. Peddie Jon. The History of Visual Magic in Computers: How Beautiful Images Are Made in CAD, 3D, VR and AR. Springer: 2013.

3. Shklyar Dmitry. 3D Rendering History. Part 1: Humble Beginnings [online]. CGSociety; 2004.
   URL: http://www.cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/3d_rendering_history_part_1._humble_beginnings
   Accessed 1 December 2016.

4. Utterson Andrew. A Computer Animated Hand [online]. Library of Congress; 2011.
   URL: https://www.loc.gov/programs/static/national-film-preservation-board/documents/computer_hand2.pdf
   Accessed 1 December 2016.

5. A Computer Animated Hand [online]. Steemit; 2016.
   URL: https://steemit.com/animation/@stino-san/-a-computer-animated-hand
   Accessed 9 December 2016.

6. Price A. David. The Making of Computer Graphics for Star Wars (Episode IV), 1977 [online]. The Pixar Touch; 2009.
   URL: http://www.pixartouchbook.com/blog/2009/11/20/the-making-of-computer-graphics-for-star-wars-episode-iv-197.html
   Accessed 2 December 2016.

7. Encyclopedia Heritage World. List of Pixar Awards and Nominations (Feature Films) [online]. Project Guthenberg Self-Publishing Press.
   URL: http://www.gutenberg.us/articles/list_of_pixar_awards_and_nominations_(feature_films)
   Accessed 5 December 2016.

8. Fronczak Tom. Top 100 Most Influential Animation Studios of All-Time [online]. Animation Career Review; 2012.
   URL: http://www.animationcareerreview.com/articles/top-100-most-influential-animation-studios-all-time?page=0,9
   Accessed 6 December 2016.

9. Niculescu Armand. The Software Used in the Making of Avatar [online]. Media Division; 2010.
   URL: http://www.media-division.com/software-used-making-of-avatar/
   Accessed 6 December 2016.

10. The 82nd Academy Awards | 2010 [online]. Oscars.com; 2010.
    URL: http://www.oscars.org/oscars/ceremonies/2010
    Accessed 9 December 2016.

11. Walker John. The Autodesk File. Bits of History, Words of Experience [online]. Index Librorum Liberorum; 2016.
URL: http://www.fourmilab.ch/autofile/
Accessed 5 December 2016.

12. Softimage: "16 Years of Leadership and Innovation" [online]. Softimage; 2001.
URL: http://web.archive.org/web/20021015000229/www.softimage.com/Corporate/Press/Facts/16years.htm
Accessed 9 December 2016.

13. Alias About Us [online]. Alias; 2004.
URL: https://web.archive.org/web/20040411045330/http://www.alias.com/eng/about/history/
Accessed 9 December 2016.

14. Elliott Phil. Autodesk to Acquire Softimage [online]. GamesIndustry.biz; 2008.
URL: http://www.gamesindustry.biz/articles/autodesk-to-acquire-softimage
Accessed 10 December 2016.

15. Autodesk to Acquire Alias for $182 Million Cash [online]. CGISociety; 2005.
URL: http://www.cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/autodesk_to_acquire_alias_for_182_million_cash
Accessed 10 December 2016.

16. Johnson Steve. Autodesk Confirms Outrageous Upgrade Price Increase [online]. Blog Nauseam; 2012.
URL: http://www.blog.cadnauseam.com/2012/10/19/autodesk-confirms-outrageous-upgrade-price-increase/
Accessed 18 December 2016.

17. Wong Kenneth. Autodesk Will Only Sell Subscription Licenses for Desktop Products After February 1, 2016 [online]. Digital Engineering; 2015.
URL: http://www.digitaleng.news/virtual_desktop/2015/02/autodesk-will-only-sell-subscription-licenses-for-desktop-products-after-february-1-2016/
Accessed 18 December 2016.

18. History [online]. Blender.org; 2013.
URL: https://www.blender.org/foundation/history/
Accessed 22 December 2016.

19. Top 25: Most Popular 3D Modeling & Design Software for 3D Printing [online]. 3D Print Pulse; 2015.
URL: http://www.3dprintpulse.com/software/?open-article-id=3886965&article-title=top-25--most-popular-3d-modeling---design-software-for-3d-printing&blog-domain=materialise.com&blog-title=i-materialise
Accessed 9 December 2016.

20. Hoffmann Vasco. A Brief History of 3D Scanning [online]. 3D Scanners Ltd; 1998.
URL: http://vr.isdale.com/3DScanners/3d_scan_history/history.htm
Accessed 10 December 2016.

21. Cyberware. Cyberware Scanners [online]. Cyberware; 1999.
URL: http://cyberware.com/products/scanners/

Accessed 10 December 2016.

22. Cyberware. Domestic Product Price [online]. Cyberware; 1999.
URL: http://cyberware.com/pricing/domesticPriceList.html
Accessed 10 December 2016.

23. Japantech. 高速3Dﾚｰｻﾞｰｽｷｬﾝ [online]. Japantech.
URL: http://www.japantech.co.jp/pc/contents15.html#%E9%A1%94%E9%A0%AD%E9%83%A8%EF%BC%93%EF%BC%A4%E3%82%B9%E3%82%AD%E3%83%A3%E3%83%B3
Accessed 1 January 2017.

24. Creaform3D. Legacy Products: REVscan Scanner [online]. Creaform3D; 2017.
URL: https://www.creaform3d.com/en/customer-support/legacy-products/revscan-scanner
Accessed 11 February 2017.

25. DeLaurentis Peter. 3D Scanning: A New Tool for Cracking Tough Cases [online]. Forensic Magazine; 2009.
URL: http://www.forensicmag.com/article/2009/02/3d-scanning-new-tool-cracking-tough-cases
Accessed 11 February 2017.

26. Noel Julien. Computed Tomography Advances for 3d Scanning [online]. Medical Design; 2009.
URL: http://medicaldesign.com/contract-manufacturing/computed-tomography-advances-3d-scanning
Accessed 11 February 2017.

27. Blecha Erika, Bowers Hannah, Barton Justin. Royal Tombs at Kasubi [online]. CyArk.
URL: http://archive.cyark.org/royal-tombs-at-kasubi-info
Accessed 11 February 2017.

28. Luebke Dr. David, Lutz Christopher, Wang Rui, Woolley Cliff. Scanning Monticello [online]. University of Virginia; 2002.
URL: http://www.cs.virginia.edu/Monticello/
Accessed 11 February 2017.

29. www.3ders.org. The Best 3D Scanners of 2015 [online]. www.3ders.org; 2015.
URL: http://www.3ders.org/articles/20151209-best-3d-scanners-2015.html
Accessed 11 February 2017.

30. Engelmann Francis. FabScan: Affordable 3D Laser Scanning of Physical Objects [online]. RWTH Aachen University; 2011.
URL: https://hci.rwth-aachen.de/materials/publications/engelmann2011a.pdf
Accessed 11 December 2016.

31. Lukas Mario. FabScan Pi - an Open-Hardware Stand-alone Web-enabled 3D Scanner [online]. RWTH Aachen University; 2015.
URL: http://hci.rwth-aachen.de/materials/publications/lukas2015a.pdf
Accessed 11 December 2016.

32. Jan Thar. 3D Scanner: FabScan Pi [online]. Instructables; 2016.

URL: http://www.instructables.com/id/3D-Scanner-FabScan-Pi/?ALLSTEPS.
Accessed 11 December 2016.

33. Lansard Martin. The Five Best (and Free!) 3D Scanning Mobile Apps [online]. Aniwaa; 2016.
URL: http://www.aniwaa.com/blog/five-best-free-3d-scanning-mobile-apps/
Accessed 8 January 2017.

34. FBI.gov. Crime Scene Documentation [online]. FBI.gov.
URL: https://www.fbi.gov/services/laboratory/forensic-response/crime-scene-documentation
Accessed 30 December 2016.

35. Lee J. Jane. 5 Ways Smithsonian Uses 3-D Scanning to Open Up History [online]. National Geographic; 2013.
URL: http://news.nationalgeographic.com/news/2013/09/130904-3d-printing-smithsonian-whale-skeleton-technology-science/
Accessed 4 December 2016.

36. Hosoi Fumiki, Nakabayashi Kazushige, Omasa Kenji. 3-D Modeling of Tomato Canopies Using a High-Resolution Portable Scanning Lidar for Extracting Structural Information [online]. MDPI; 2011.
URL: http://www.mdpi.com/1424-8220/11/2/2166/htm
Accessed 11 December 2016.

37. Mode Lab. 1.6.1 What is a Mesh? [online]. Mode Lab; 2015.
URL: http://grasshopperprimer.com/en/1-foundations/16/1_What%20is%20a%20Mesh.html
Accessed 24 December 2016.

38. Sculpteo. 3D Model and CAD Model [online]. Sculpteo.
URL: https://www.sculpteo.com/en/glossary/3d-model-definition/
Accessed 11 December 2016.

39. Sculpteo. 3D Modeling: Creating 3D Objects [online]. Sculpteo.
URL: https://www.sculpteo.com/en/glossary/3d-modeling-definition/
Accessed 11 December 2016.

40. A Mostafa. 3D Laser Scanners' Techniques Overview [online]. International Journal of Science and Research; 2015.
URL: https://www.ijsr.net/archive/v4i10/SUB158346.pdf
Accessed 12 December 2016.

41. Munaro Matteo, Wai Yan So Edmond, Tonello Stefano, Menegatti Emanuele. Efficient Completeness Inspection Using Real-Time 3D Color Reconstruction with a Dual-Laser Triangulation System [online]. ResearchGate; 2015.
URL: https://www.researchgate.net/publication/283108894_Efficient_Completeness_Inspection_Using_Real-Time_3D_Color_Reconstruction_with_a_Dual-Laser_Triangulation_System
Accessed 25 December 2016.

42. 3D Systems, Inc. 3D Scanners. A Guide to 3D Scanner Technology [online]. 3D Systems, Inc.
URL: http://web.archive.org/web/20161017220639/http://www.rapidform.com/

3d-scanners/
Accessed 26 December 2016.

43. McDonald Kyle. Structured Light 3D Scanning [online]. Instructables.
URL: http://www.instructables.com/id/Structured-Light-3D-Scanning/
?ALLSTEPS
Accessed 26 December 2016.

44. Eren Gönen. 3D Scanning of Transparent Objects [online]. TEL; 2011.
URL: https://tel.archives-ouvertes.fr/file/index/docid/584061/filename/
these_A_EREN_Gonen_2010.pdf
Accessed 26 December 2016.

45. ShapeGrabber. Inspecting Transparent or Reflective Parts with a 3D Laser Scanner [online]. ShapeGrabber; 2015.
URL: http://www.shapegrabber.com/
inspecting-transparent-reflective-parts-3d-laser-scanner/
Accessed 26 December 2016.

46. Wohl Jimmy. Digitizer Education | Part 3: Materials for 3D Scanning [online]. MakerBot; 2013.
URL: https://www.makerbot.com/media-center/2013/11/08/
digitizer-education-part-3-materials-for-3d-scanning
Accessed 26 December 2016.

47. Walford Alan. What is Photogrammetry? [online]. Photogrammetry; 2007.
URL: http://www.photogrammetry.com/
Accessed 1 March 2017.

48. INFINITE-REALITIES. Next Generation Photometric Scanning [online]. INFINITE-REALITIES; 2016.
URL: http://ir-ltd.net/next-gen-photometric-scanning/
Accessed 12 December 2016.

49. Zhang Yiwei, Gibson M. Graham, Hay Rebecca, Bowman W. Richard, Padgett J. Miles, Edgar P. Matthew. A Fast 3D Reconstruction System with a Low-cost Camera Accessory [online]. Scientific Reports; 2015.
URL: http://www.nature.com/articles/srep10909
Accessed 26 December 2016.

50. Statista. Number of Smartphone Users Worldwide from 2014 to 2020 (in Billions) [online]. Statista; 2016.
URL: https://www.statista.com/statistics/330695/
number-of-smartphone-users-worldwide/
Accessed 14 January 2017.

51. Triggs Robert. How Far We've Come: A Look at Smartphone Performance over the Past 7 Years [online]. Android Authority; 2015.
URL: http://www.androidauthority.com/
smartphone-performance-improvements-timeline-626109/
Accessed 22 January 2017.

52. IDC. Smartphone OS Market Share, 2016 Q3 [online]. IDC; 2016.
URL: http://www.idc.com/promo/smartphone-market-share/os

Accessed 14 January 2016.

53. Android Developers. Dashboards [online]. Android Developers; 2017.
URL: https://developer.android.com/about/dashboards/index.html
Accessed 14 January 2017.

54. PhoneArena.com. Did You Know How Many Different Kinds of Sensors Go Inside a Smartphone? [online]. PhoneArena.com; 2014.
URL: http://www.phonearena.com/news/Did-you-know-how-many-different-kinds-of-sensors-go-inside-a-smartphone_id578
Accessed 14 January 2017.

55. Stonekick Apps. Magnetometers, Accelerometers, and the Calibration Procedure for Your Android Device [online]. Stonekick Apps; 2013.
URL: http://www.stonekick.com/blog/magnometers-accelerometers-and-calibrating-your-android-device/
Accessed 2 February 2017.

56. Bjork Gail. What Is a Megapixel? [online]. Digicamhelp.
URL: http://digicamhelp.com/camera-features/camera-parts/megapixels/
Accessed 22 January 2017.

57. Lee Paul, Stewart Duncan. Predictions 2016. Photo Sharing: Trillions and Rising [online]. Deloitte; 2015.
URL: https://www2.deloitte.com/global/en/pages/technology-media-and-telecommunications/articles/tmt-pred16-telecomm-photo-sharing-trillions-and-rising.html#
Accessed 22 January 2017.

58. Patkar Mihir. Which Smartphone Has the Best Autofocus; How Does It Work? [online]. MakeUseOf; 2015.
URL: http://www.makeuseof.com/tag/which-smartphone-has-the-best-autofocus-how-does-it-work/
Accessed 22 January 2017.

59. ini4j. Java API for Handling Windows Ini File Format [online]. Sourceforge.net; 2011.
URL: http://ini4j.sourceforge.net/
Accessed 21 January 2017.

60. BoofCV. Relative Speed of BoofCV and OpenCV [online]. BoofCV; 2013.
URL: http://boofcv.org/index.php?title=Performance:OpenCV:BoofCV
Accessed 22 January 2017.

61. OpenCV. About [online]. OpenCV; 2017.
URL: http://opencv.org/about.html
Accessed 22 January 2017.

62. Android Developers. Position Sensors [online]. Android Developers.
URL: https://developer.android.com/guide/topics/sensors/sensors_position.html
Accessed 31 January 2017.

63. Android Developers. SensorEvent [online]. Android Developers.
URL: https://developer.android.com/reference/android/hardware/

SensorEvent.html
Accessed 12 February 2017.

64. Android Developers. Sensors Overview [online]. Android Developers.
URL: https://developer.android.com/guide/topics/sensors/sensors_overview.html
Accessed 21 February 2017.

65. Jones M. Eric, Fjeld Paul. Gimbal Angles, Gimbal Lock, and a Fourth Gimbal for Christmas [online]. NASA; 2000.
URL: https://www.hq.nasa.gov/alsj/gimbals.html
Accessed 21 February 2017.

66. Android Developers. SensorManager [online]. Android Developers.
URL: https://developer.android.com/reference/android/hardware/SensorManager.html
Accessed 21 February 2017.

67. GIMP. The Free & Open Source Image Editor [online]. GIMP.
URL: https://www.gimp.org/
Accessed 16 January 2017.

68. Brian Matt. Google's New 'Material Design' UI Coming to Android, Chrome OS and the Web [online]. Engadget; 2014.
URL: https://www.engadget.com/2014/06/25/googles-new-design-language-is-called-material-design/
Accessed 22 January 2017.

69. Material Design. Introduction [online]. Material design.
URL: https://material.io/guidelines/
Accessed 16 January 2017.

70. Ferreira Michel. Would You Like Fries with That? [online]. Booking.com Blog; 2014.
URL: https://blog.booking.com/hamburger-menu.html
Accessed 24 January 2017.

71. FSymbols. Tick Symbols [online]. FSymbols; 2017.
URL: http://fsymbols.com/signs/tick/
Accessed 24 January 2017.

72. Veltri Alejandro. Why Are the “Undo” and “Redo” Arrow Icons Commonly Round? [online]. User Experience Stack Exchange; 2015.
URL: http://ux.stackexchange.com/questions/83723/why-are-the-undo-and-redo-arrow-icons-commonly-round
Accessed 24 January 2017.

73. Turner Julia. The Big Red Word vs. the Little Green Man [online]. Slate; 2010.
URL: http://www.slate.com/articles/life/signs/2010/03/the_big_red_word_vs_the_little_green_man.html
Accessed 24 January 2017.

74. Bolshakova Anastasya. Media, Multimedia, Player, Square, Stop [online]. Iconfinder.

URL: https://www.iconfinder.com/icons/1167977/
media_multimedia_player_square_stop_stopping_icon#size=1
Accessed 24 January 2017.

75. Bolshakova Anastasya. Check, Checklist, Dots, List, Menu Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/1167996/
check_checklist_dots_list_menu_icon#size=1
Accessed 24 January 2017.

76. Designerz Base. Available, Checkmark, Done Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/186405/
available_checkmark_done_icon#size=1
Accessed 24 January 2017.

77. Lung Yannick. Undo Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/183192/undo_icon#size=1
Accessed 24 January 2017.

78. Google. App, Exit, To Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/326635/app_exit_to_icon#size=1
Accessed 24 January 2017.

79. Silva Thiago. Basic, Edit, Pen, Pencil, Thiago Pontes Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/981079/
basic_edit_pen_pencil_thiago_pontes_icon#size=1
Accessed 24 January 2017.

80. Lung Yannick. Gauge, Pressure, Reading Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/183415/
gauge_pressure_reading_icon#size=1
Accessed 24 January 2017.

81. Blender. Free and Open 3D Creation Software [online]. Blender.
URL: https://www.blender.org/
Accessed 22 January 2017.

82. Rujikietgumjorn Sitapa. Segmentation Methods for Multiple Body Parts [online]. The University of Tennessee; 2008.
URL: https://www.imaging.utk.edu/publications/papers/dissertation/
2008-aug-sitapi-pilot.pdf
Accessed 30 January 2017.

83. Liu Dingding, Soran Bilge, Petrie Gregg, Shapiro Linda. A Review of Computer Vision Segmentation Algorithms [online]. University of Washington; 2012.
URL: https://courses.cs.washington.edu/courses/cse576/12sp/notes/remote.pdf
Accessed 29 January 2017.

84. Marsh Matthew. "GrabCut" - Interactive Foreground Extraction using Iterated Graph Cuts [online]. Rhodes University; 2005.
URL: http://www.cs.ru.ac.za/research/g02m1682/
Accessed 19 February 2017.

85. OpenCV. Miscellaneous Image Transformations [online]. OpenCV; 2015.

URL: http://docs.opencv.org/3.1.0/d7/d1b/
group__imgproc__misc.html#ga909c1dda50efcbeaa3ce126be862b37f
Accessed 12 February 2017.

86. Google. Favorite, Rate, Star Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/326703/favorite_rate_star_icon
Accessed 4 February 2017.

87. Xinh Studio. Camera, Device, Film, Movie, Multimedia, Recorder, Video Icon [online]. Iconfinder.
URL: https://www.iconfinder.com/icons/763252/
camera_device_film_movie_multimedia_recorder_video_icon
Accessed 4 February 2017.

88. Matplotlib. Introduction [online]. Matplotlib.
URL: http://matplotlib.org/
Accessed 3 February 2017.

89. 3DXO.com. Christmas Tree Star 2 [online]. 3DXO.com.
URL: http://www.3dxo.com/models/10580_christmas_tree_star_2
Accessed 3 February 2017.

90. OpenCV. Canny Edge Detector [online]. OpenCV; 2017.
URL: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/
canny_detector/canny_detector.html
Accessed 4 February 2017.

91. OpenCV. Basic Structures [online]. OpenCV; 2014.
URL: http://docs.opencv.org/2.4.10/modules/core/doc/basic_structures.html
Accessed 4 February 2017.

92. Olsson Karin, Persson Therese. Shape from Silhouette Scanner [online]. Digitala Vetenskapliga Arkivet; 2001.
URL: http://www.diva-portal.org/smash/get/diva2:18671/FULLTEXT01.pdf
Accessed 18 February 2017.

93. Owen Scott. 3D Rotation [online]. ACM SIGGRAPH; 1998.
URL: https://www.siggraph.org/education/materials/HyperGraph/
modeling/mod_tran/3drota.htm
Accessed 16 February 2017.

94. Bourke Paul. Points, Lines, and Planes [online]. Paul Bourke - Personal Pages; 1988.
URL: http://paulbourke.net/geometry/pointlineplane/
Accessed 18 February 2017.

95. Bourke Paul. Object Files (.obj) [online]. Paul Bourke - Personal Pages.
URL: http://paulbourke.net/dataformats/obj/
Accessed 15 February 2017.