UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA INFORMATICA
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA MAGISTRALE

A LOW-COST, EFFICIENT AND ACCURATE HUMAN BODY SCANNER

RELATORE: Ch.mo Prof. Enoch Peserico Stecchini Negri De Salvi

CORRELATORE: Ing. Federica Bogo

LAUREANDO: *Christian Piccolo*

Anno Accademico 2011-2012

Twenty years from now you will look back more regretfully upon the things you didn't do than those you did. So set free the bowlines, sail away from safe harbor. Dream. Explore. Discover.

Mark Twain

To my family,
for coloring my life with love and laughter.

Abstract

Following the emergence of low-cost, high quality cameras and projectors, 3D scanners are becoming more and more affordable, making way for a number of different industries ranging from clothing to mechanical and film. Advancements in 3D human body scanning offer even greater potential for healthcare applications by transforming our ability to accurately measure a person's body size, shape, and skin-surface area, in order to track any changes and foresee the development of disease or ailment. This thesis describes our approach in building a structured light 3D scanner to help dermatologists track the evolution of moles and psoriasis of their patients.

Acknowledgements

I am grateful to my adviser Enoch Peserico for his many years of guidance and support. He has been an influential person in my academic career, responsible for kindling my interest in computer science and for giving me the opportunity to develop this project. I feel privileged for having had Federica Bogo as my assistant adviser. She helped me for countless hours in the development of this thesis by providing valuable assistance and advice.

I extend my gratitude to my housemates Matteo Vanin, Nicola Belli, Paolo Anzelini and Giacomo Borz for putting up with me during these last eight months. I will never forget the good time we had in France and Africa. I would like to thank my college advisor Martina Muscarella for her constant guidance concerning my approach to my courses. I am also very grateful to Ioannis Moukazis and Eleonora Lozzi for all the laughter and food we shared together during these five years. I would also like to thank all the friends I made at UCSB for being an essential part of the craziest year of my life. It could not have been the same without you. To Davide Tortora, thank you for agreeing to host me once again in California next year. Finally, I am grateful to all of my friends for all the memorable times we had together. You have always been, and remain, an important part of my life.

Last, but certainly not least, I thank my family for believing in me throughout my studies and for supporting me in my years-long foray into America. You have always pushed me to sail away from the safe harbor and explore the world. Thank you mamma, papà and Alexa, I am truly grateful.

Contents

Chapter 1

Introduction

The last decade has seen a major technological improvement in diagnostic studies. In the present climate, it is possible to reveal detailed information about the body's internal structure using X-rays, magnetic resonance imaging (MRI), computed tomography (CT scan), and ultrasound. These technologies are very helpful to medical professionals in the study of physiology and anatomy in vivo as well as in the diagnosis and monitoring of a myriad of disease states.

For these ends, external measurements can be as useful as internal scanning. Doctors widely use information about the patient's body size and shape to assess nutritional status and developmental normality, as well as to calculate the requirements of drug, radiotherapy, and chemotherapy dosages and the production of prostheses. Medical research and practice can harness the power of 3D scanners to make a large impact on the epidemiological study of many diseases, surpassing the current confidence in the body mass index (BMI), which is used to quantify many traits without accurately quantifying any. Finally, the ability to obtain such information could considerably improve physiological research by contributing to comprehensive regional indices of people's size, as required by such research, as well as guide clinical practice in hospitals and community clinics.

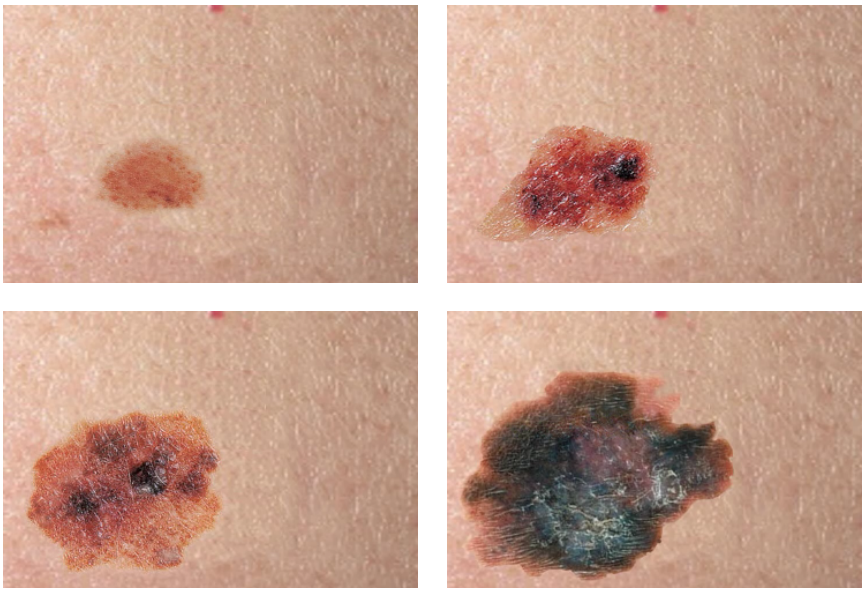


Figure 1.1: *Progression of an atypical mole to radial melanoma.*

The potential of 3D human body scanners relies on their capability to capture accurate 3D point clouds in a matter of minutes. Afterwards, a computer automatically extracts surface details and maps high resolution textures. It is then possible to extract hundreds of measurements from the 3D model while eliminating manual measurement and transcription errors, providing a comparison for future measurements of the patient, and greatly reducing the cost of anthropometric surveys.

It is greatly important to note that a 3D human body scanner does not pose any health risk to the patient, as it is built using only a camera and projector.

Generally, a person has his or her moles examined only once a year, if not less. A fair percentage of the population has never had their moles examined in a lifetime; however, nodular melanomas can spread internally in as little as three months, and most radial melanomas can spread internally within 6 to 18 months from the first noticeable change of a pre-existing mole or the

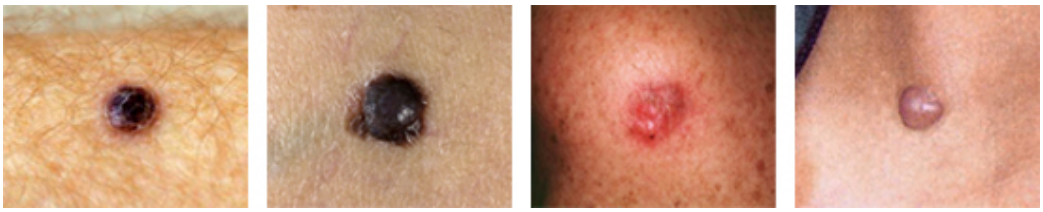


Figure 1.2: *Four different nodular melanomas.*

Figure 1.3: *Psoriasis on the back.*

appearance of a new mole (Figure 1.1) [Mel][NSM+94]. Once the melanoma spreads internally, curability is only about 50 percent and decreases quickly as the thickness of the mole further increases. The capacity to understand the growth rate of lesions caused by psoriasis is also an extremely important aid to finding the right cure for the patient.

The primary goal of our project is to create a 3D human body scanner to help dermatologists track the evolution of moles and psoriasis on their patients, as shown in Figure 1.2 and Figure 1.3.

The outline of the thesis is the following. Chapter 2 is devoted to introduce an overview on our project's goals. Moreover, it covers the state-of-the-art about 3D scanning technologies. Chapter 3 describes the basic mathematical background for 3D scanners, with particular regard to the camera and projector mathematical models. Chapter 4 lists the hardware and software we have used to build the 3D scanner. Chapter 5 and 6 are the crucial part of the thesis and they explore in great detail our human body scanner application.

Afterwards, Chapter 7 discusses the results and compares our human body scanner with a professional one. Finally, Chapter 8 and 9 talk about some promising directions for future research and they draw the conclusion of our work.

Chapter 2

3D Scanning Technology: an Overview

Understanding the great impact that a human body scanner could have on medical research, and in collaboration with the Dermatology Unit at the Department of Medical and Surgical Specialties of the University of Padova and the Department of Information Engineering the University of Padova, we have designed a human body scanner to fulfill the following goals:

Accuracy The scanner must be very precise, because moles can be smaller than a few millimeters. Furthermore, we should be able to scan close to 100% of the patient's skin surface in order to properly analyze all possible moles and the psoriasis growths.

Efficiency The time needed for a complete scan should be as minimal as possible. Ideally, people should be scanned more than once a year to check on their status; therefore, our scanner should take at most 5 minutes to build the entire human model.

Costs Today, the cost of 3D scanners capable of capturing an entire human body ranges from 50K to 300K+ dollars. We want to build an affordable scanner that costs less than $10,000.

Our final mission is to provide hospitals with this scanner, in order to check patients at least once every two months with an automated scan. Only in the instance of an identified abnormality will a doctor step in to personally examine the patient. Once a year, the patient will have the usual face-to-face meeting and examination with a dermatologist.

Finally, it is necessary to emphasize that our human body scanner does not replace the role of a dermatologist. Instead, it enhances medical possibilities by tracking the evolution of the patient's skin, more frequently than the general practice of yearly checks.

In order to build a great human body scanner, we had to study the state-of-the-art technologies available nowadays. There are many different technologies that can be used to build a 3D scanner, each with specific benefits and drawbacks. This Chapter describes the most important 3D scanner typologies that are in use today to explain the choice we have made in building our human body scanner.

Figure 2.1 shows a detailed taxonomy about 3D scanners. The first distinction is between *contact* and *non-contact* scanners. Contact scanners probe the surface of the object and are therefore able to reconstruct its shape. One example is a Coordinate Measuring Machine (CMM), which is used primarily in manufacturing and can be very accurate; however, the biggest disadvantage of a CMM, and generally of any contact scanner, is that the probe of the scanner may ultimately modify durable objects and even destroy very fragile ones. Finally, contact 3D scanners require a very precise mechanism to move the probing arm and are unable to scan objects that cannot remain completely still, like human beings.

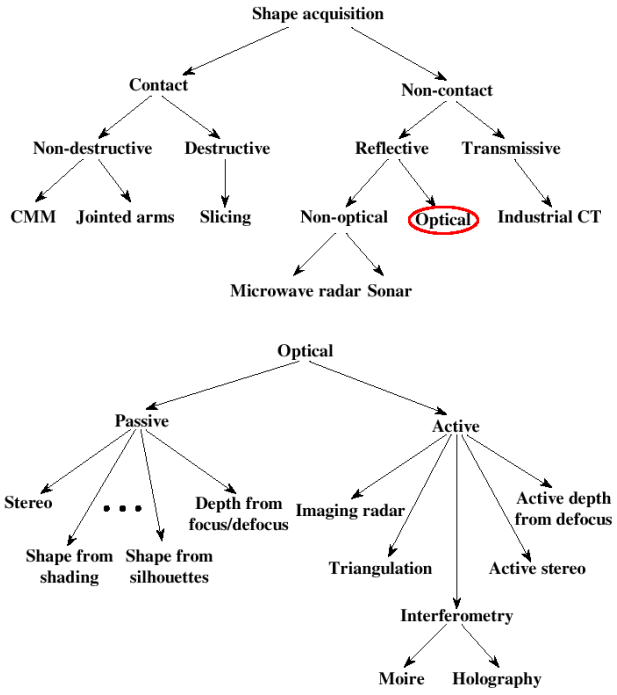


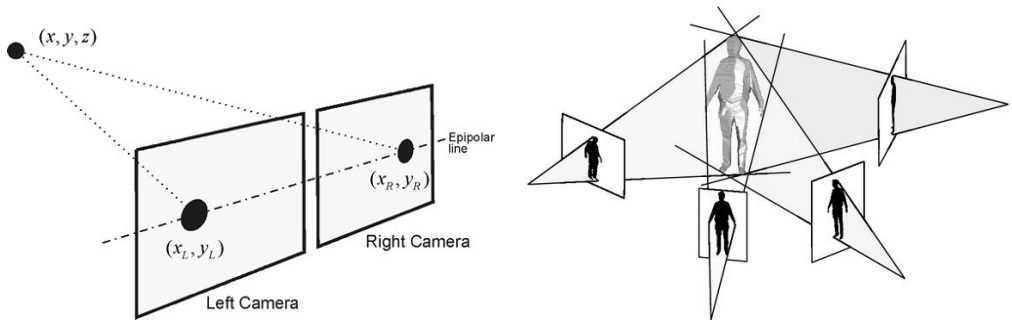
Figure 2.1: *A taxonomy of different 3D scanning technologies.*

The branch of non-contact scanners has many subclasses, but we will be focusing on one based on the adoption of optical techniques to observe and analyze an object. The subclass of *optical* scanners.

The category of optical scanners can be further divided into *active* and

(a) Dea One CMM (b) Probe

Figure 2.2: *A Coordinate Measuring Machine and a zoom on its probe.*



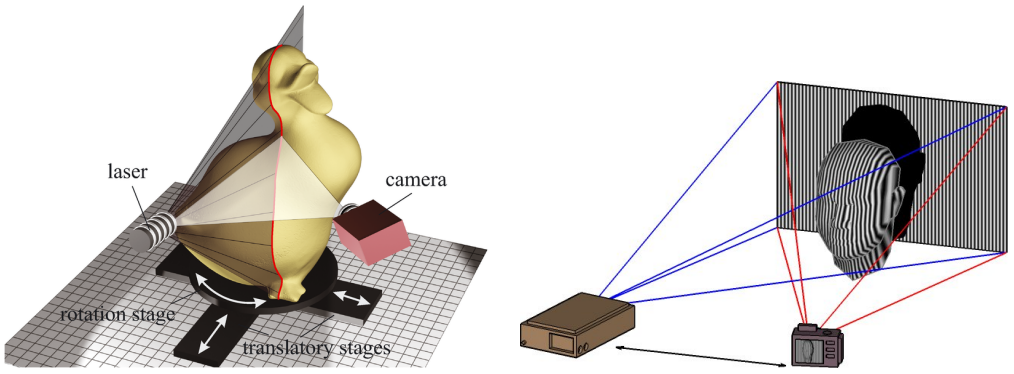
(a) Stereo Vision (b) Shape from Silhouette

Figure 2.3: *Two of the most common ways to realize a passive optical 3D scanner.*

passive. The following Sections will describe these two very similar techniques, which have become mainstream technologies for building cheap and high quality scanners.

2.1 State-Of-The-Art Optical Scanners

Passive Optical Scanners Passive scanners do not require any additional light source. The most widely used passive scanners are *stereoscopic scanners* and *shape from silhouette scanners*. The first ones require the use of two calibrated cameras to take pictures of the same object from two different angles. First the 2D projection of a given point is identified in both pictures, and then a simple triangulation algorithm recovers the depth of that point (Figure 2.3(a)). The idea is powerful, simple and mirrors the human visual system, however it contains some major drawbacks concerning surfaces that pose difficulty for the correspondence of the same point in different pictures. For example, human skin cannot be modeled using a stereoscopic scanner because it is generally flat and prevents robust matching. For this reason, multi-view stereoscopic scanners are not able to produce accurate and reliable 3D measurement of a naked person.



(a) Laser Scanner (b) Structured Light Scanner

Figure 2.4: *Two of the most common ways to realize an active optical 3D scanner.*

The final passive technology we will mention is the *shape from silhouette* [Lau94] [Lau95]. It relies on the ease of separating an object from its background, and thus through the use of many calibrated cameras surrounding the object (or alternatively one camera and a turntable), it is possible to create a visual hull of the object, i.e: the union of the generalized viewing cones defined by each camera's center of projection and the detected silhouette boundaries, as evident in Figure 2.3(b). The drawback of this technique is the need for either a large quantity of cameras or a very precise turntable to reconstruct a 3D model. Moreover, it is impossible to detect convex surfaces like the belly button or the area between a woman's breasts with this method alone.

Active Optical Scanners Active scanners use an external light source to overcome the problem of detecting point correspondence in two separate pictures. The external light source produces a structured illumination on the object to be scanned, and the sensor, which is typically a CCD camera, acquires the images of the distorted pattern reflected by the object's surface. In most cases, the depth information is reconstructed by triangulation, given the known relative positions of the light source and camera.

The two most common external light sources are a *coherent light* (e.g. a laser-beam) or an *incoherent light* (e.g. a projector). When laser scanners first became popular, the usual way to create a 3D model was to move a laser point across the object surface and reconstruct it point-by-point. This method, however, was painstakingly slow. Following the development of low-cost, high-quality CCD arrays, *slit scanners* emerged as a powerful alternative. In this design, a laser projector creates a single planar sheet of light. This "slit" is then mechanically-swept across the surface, and the registered deflection of the laser source defines a 3D plane. The depth is recovered by the intersection of this plane with the set of lines passing through the 3D stripe on the object's surface and the camera's center of projection (Figure 2.4(a)).

Currently, a digital *structured light* projector can be used to eliminate the mechanical motion required to translate the laser strip across the surface. To take full advantage of the projector's power, which is capable of displaying arbitrary color images, structured lighting sequences have been developed in order to assign the projector-camera correspondences in relatively few frames. In general, the identity of each plane can be encoded spatially (i.e., within a single frame, like color patterns) or temporally (i.e., across multiple frames, like the gray codes in Figure 2.4(b)) or phase shifting codes), or with a combination of both spatial and temporal encodings. There are benefits and drawbacks to each strategy. For instance, purely spatial encodings allow the use of a single static pattern for reconstruction, thus enabling dynamic scenes to be captured. Alternatively, purely temporal encodings are more likely to benefit from redundancy, reducing reconstruction artifacts. We refer the reader to a comprehensive assessment of such codes by Salvi et al. [SPB04].

2.2 Project Requirements

In the design of our affordable, efficient and accurate human body scanner, we have to respect some requirements. For example, we should avoid touching the patient, but must assure that the patient remains standing or lying still. Moreover, we still need to respect the goals that we described at the beginning of this Chapter: accuracy, efficiency, and economy. The first requirement listed here requires the exclusion of contact scanner technology; thus, the remaining choice was between a passive or active optical scanner. Because passive scanners are not robust with flat skin surfaces or concavities, we had to decide whether to use a laser or a projector to implement our active optical scanner.

Finally, we chose to build a structured light 3D scanner with a projector for the following reasons:

Accuracy While not as precise as a laser scanner, a projector still allows for the detection of body changes < 1 mm.

Efficiency This type of scanner is much faster than a laser scanner, thanks to its multiple projected patterns. In Chapter 7, we will compare the speed of a professional laser scanner with the one of a our structured light scanner.

Costs A projector is much cheaper than a laser, because it does not require an expensive and precise mechanical turntable.

Health Risk It is much more dangerous to project a laser onto a face or eyes than it is to simply use a light source; nonetheless, covering the patient's eyes will still be fundamental to ensure enough safety.

2.3 Project Development Steps

This project will be divided into four steps:

- reconstruction of the 3D model for a patient's body's part;
- mapping the texture on the surface of the model;
- merging all the models together to generate a complete human body model for the patient;
- registration of each different model taken at a different time;
- development of algorithms to analyze the models and detect suspicious changes on the patient's skin.

This thesis covers the first two steps.

Beginning in the following Chapter, we will delve into the theoretical and practical aspects of our structured light - human body scanner.

Chapter 3

Camera and Projector Mathematical Models

The goal of our project is to build a 3D model of a body in the most efficient way possible. Our method involves projecting a known pattern onto the body and capturing the resulting illuminated image with a camera. In order to do this, we need to use a mathematical model of a camera and projector that allows for the reconstruction of a 3D shape by geometric triangulation.

First, it is necessary to understand the image formation process and how to implement triangulation using only the pattern projected on the subject's surface and a picture taken from a camera. Then, we will use basic algebra in order to create a complete mathematical model that describes our world.

3.1 Geometry

Notation

Throughout this thesis we will use the following mathematical notation:

- vectors are taken as real vectors with real value coordinates $\mathbf{v} \in \mathbb{R}^{3\times 1}$ and their length is a scalar $\|\mathbf{v}\| \in \mathbb{R}$;

- $\mathbf{v}^t \in \mathbb{R}^{1\times 3}$ is a row vector resulting from transposing the column vector $\mathbf{v}$;

- matrix multiplication is used to compute the inner product $\mathbf{v_1}^t\mathbf{v_2} \in \mathbb{R}$ of two vectors $\mathbf{v_1}$ and $\mathbf{v_2}$. The result corresponds to a scalar, whose value is $\|\mathbf{v_1}\|\|\mathbf{v_2}\|\cos(\alpha)$. Here α denotes the angle formed by the two vectors $(0 \leq \alpha \leq 180)$;

- the vector product $\mathbf{v_1} \times \mathbf{v_2} \in \mathbb{R}^3$ is a vector perpendicular to both $\mathbf{v_1}$ and $\mathbf{v_2}$ of length $\|\mathbf{v_1} \times \mathbf{v_2}\| = \|\mathbf{v_1}\|\|\mathbf{v_2}\|\sin(\alpha)$ and direction determined by the right hand rule. It is worthwhile to point out that the vector product of two linearly dependent vectors is equal to zero.

Points and Vectors

While points simply describe a location in our 3D world, vectors have no fixed position in space and have both magnitude and direction.

We utilize an *affine space* to understand our mathematical model; this space is made up of a set of points P and a vector space V. The vector space V implies that vectors can be added to each other and multiplied by any scalar. Points and vectors of this space are related in a fixed, but simple manner by the following axioms:

1. a point plus a vector is another point:

$$p + \mathbf{v} = q$$

2. the difference between two points is a vector:

$$p - q = \mathbf{v}$$

3. if p is a point, $\mathbf{v}$ is a vector and λ is a scalar then $p + \lambda\mathbf{v} = q$ is another point;

4. an affine combination of N point $\lambda_1 p_1 + \cdots + \lambda_N p_N$, with $\lambda_1 + \cdots + \lambda_N = 1$, is well defined:

$$\lambda_1 p_1 + \lambda_2 p_2 + \cdots + \lambda_N p_N = \lambda_2(p_2 - p_1) + \cdots + \lambda_N(p_N - p_1)$$

Lines and Rays

In the previous Section we saw that adding a vector multiplied by a scalar to a point generates another point: $p + \lambda\mathbf{v} = q$. We immediately see that by changing the value of λ, we are able to generate an infinite number of points that all lie in the same line L, defined as:

$$L = \{p = q + \lambda\mathbf{v} : \lambda \in \mathbb{R}\}$$

In this definition λ can be positive or negative. We therefore see that the starting point q can be replaced by any other point laying in the same line L. If we force λ to assume only positive values then we no longer have a line but instead, a ray:

$$R = \{p = q + \lambda\mathbf{v} : \lambda \in \mathbb{R}, \lambda \geq 0\}$$

In a ray, q is of utmost importance because it defines the origin of the ray and cannot change without altering the entire structure of the ray.

Planes

In the same way that we mathematically describe a line, we are also able to represent a plane P. We need only to know a point p and two linearly independent vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ lying on the plane. In the following two paragraphs we are going to demonstrate two different ways to mathematically describe a plane; both of these will be very useful in our program for the reconstruction of our 3D model of a body.

Parametric Form The simplest and most intuitive way to describe a plane is the following:

$$P = \{p = q + \lambda_1 \mathbf{v_1} + \lambda_2 \mathbf{v_2} : \lambda_1, \lambda_2 \in \mathbb{R}\}$$

This representation is not unique because we can replace the point p with any other point lying on the plane. In the same way, we can substitute the two vectors with two new vectors, as long as they are still linearly independent and also lie on the plane P.

Implicit Form We will later see that a more useful way to describe a plane P is by using an implicit form, therefore describing the plane as the set of zeros of a linear equation in three variables. Geometrically, a point p belongs to plane P if and only if the vector $p - q$ and a vector $\mathbf{n}$ normal to the plane are orthogonal:

$$P = \left\{p : \mathbf{n}^t(p - q) = 0\right\}$$

As usual, this representation is not unique and we are able to replace both the point q and the normal vector $\mathbf{n}$.

Finally, the conversion from a parametric representation of a plane P to an implicit one is straightforward. To convert from parametric to implicit form we need only compute $\mathbf{n} = \mathbf{v_1} \times \mathbf{v_2}$; alternately, to convert from implicit to parametric form, we have to find a vector $\mathbf{v_1}$ orthogonal to $\mathbf{n}$. We can then find $\mathbf{v_2}$ by computing $\mathbf{v_2} = \mathbf{n} \times \mathbf{v_1}$.

Implicit Representation of Lines

As with planes, lines may also be described using implicit form. More specifically, they can be described as the intersection of two planes that are not parallel:

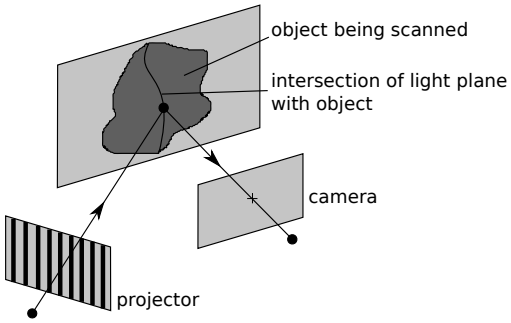


Figure 3.1: *Line-Plane Triangulation.*

$$L = \left\{ p : \mathbf{n_1}^t (p - q) = \mathbf{n_2}^t (p - q) = 0 : \mathbf{n_1} \times \mathbf{n_2} \neq 0 \right\}$$

The only condition that must be met is that $\mathbf{n_1} \times \mathbf{n_2} \neq 0$. This implies that vectors $\mathbf{n_1}$ and $\mathbf{n_2}$ should be linearly independent. This guarantees that the two planes will indeed intersect.

3.2 Line-Plane Triangulation

So far we have covered the basis of understanding how to express points, vectors, lines, rays and planes. Using only this knowledge, it is at least ideally possible to reconstruct our 3D model. In fact, as we will see in the following Chapters, it is very common for the triangulation process to project a highly identifiable pattern on the world, as well as to use pictures of the illuminated subject taken with the camera to reconstruct its shape (Figure 3.1). This is possible because the intersection of a ray of light from the projector with the subject is seen as a single illuminated point, whereas the intersection of a plane of light with the subject is seen as multiple curved segments, each of which is composed by many illuminated points.

At this point we assume that we know the position of the camera and the projector with respect to the global coordinate system (Chapter 5 will explain

how we calculate this). Under this assumption we are able to reconstruct the depth of a single illuminated point by intersecting the plane of light emanating from the projector with the ray of light hitting the camera sensor.

The math behind the triangulation process is rather simple. First we represent a line using its parametric form

$$L = \{p = q_L + \lambda \mathbf{v} : \lambda \in \mathbb{R}\}$$

and the plane using its implicit form

$$P = \left\{p : \mathbf{n}^t(p - q_P) = 0\right\}$$

Initially, we need to confirm that the line is not parallel to the plane, otherwise there would be no intersection between the two. This is done by verifying beforehand that $\mathbf{n}^t\mathbf{v} \neq 0$. Moreover, we know that the intersection point p between the line and the plane is definitely part of the line and can therefore be expressed as:

$$p = q_L + \lambda_{SOL}\mathbf{v}$$

It is now necessary to find the correct value of λ_{SOL}. This is done by intersecting the line L with the plane P, thus solving the equation:

$$\mathbf{n}^t(p - q_P) = \mathbf{n}^t(q_L + \lambda_{SOL}\mathbf{v} - q_P) = 0$$

By solving this equation for λ_{SOL} we know that:

$$\lambda_{SOL} = \frac{\mathbf{n}^t(q_P - q_L)}{\mathbf{n}^t\mathbf{v}}$$

This expression is well defined because we verify beforehand that $\mathbf{n}^t\mathbf{v} \neq 0$.

3.3 The Pinhole Model

Up until this Section, we present a mathematical model that works perfectly well in a coordinate-free description of triangulation. In practice, however, the ray of light that hits the camera sensor is saved in a discrete unit called *pixel*. Furthermore, other factors like focal length, non-square pixel and tilted image plane need to be considered in our real mathematical model in order for it to work in the reconstruction of the 3D model.

A general, simple and popular model for a camera is the *pinhole model*. In the pinhole model, the camera is described with a point o, called the *center of projection*, and an *image plane* $P = \{p = q + u_1\mathbf{v_1} + u_2\mathbf{v_2} : u_1, u_2 \in \mathbb{R}\}$, where the point q and the vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ define a local coordinate system. Every 3D point p, excluding the center of projection, has coordinates $(p^1, p^2, p^3)^t$ and determines a unique line passing through the center of projection: $p = o + \lambda\mathbf{v}$. If the line is not parallel to the image plane, then it must intersect it in a single *image point* that has coordinates u^1 and u^2, which can be written as a 3D vector $\mathbf{u} = (u^1, u^2, 1)$. Using this notation a point p can be expressed as:

$$\begin{pmatrix} p^1 \\ p^2 \\ p^3 \end{pmatrix} = [v_1|v_2|q] \begin{pmatrix} u^1 \\ u^2 \\ 1 \end{pmatrix}$$

The mathematical term for this mapping from 3D points to 2D points is called *perspective projection*. The geometry of a projector can be described with the same pinhole model we use to describe a camera; the only difference is that for a projector the light travels from the center of projection through the image plane into the world rather than vice versa (Figure 3.2).

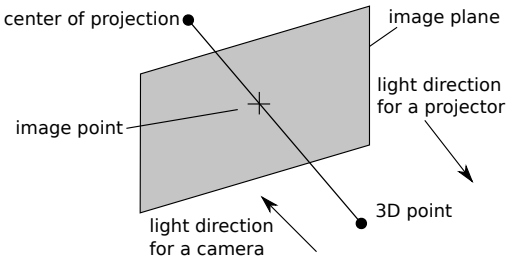


Figure 3.2: *Perspective projection under the pinhole model.*

3.3.1 The Ideal Pinhole Camera

In the ideal pinhole model the center of projection o is located at the origin of the world coordinate system $(0,0,0)^t$ and the point q and the vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ are defined as:

$$[v_1|v_2|q] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

There exists a set of 3D points that do not project onto the image plane. All these points have coordinates $(p^1, p^2, 0)^t$, i.e. $p^3 = 0$; if $p^3 \neq 0$ then the point is part of the image plane with coordinates:

$$\begin{pmatrix} u^1 \\ u^2 \\ 1 \end{pmatrix} = \begin{pmatrix} p^1/p^3 \\ p^2/p^3 \\ 1 \end{pmatrix}$$

This relation between a 3D point and its 2D projection can be expressed by saying that a 3D point can project itself onto the image plane if and only if there exists a scalar $\lambda = p^3$ such that:

$$\lambda \begin{pmatrix} u^1 \\ u^2 \\ 1 \end{pmatrix} = \begin{pmatrix} p^1 \\ p^2 \\ p^3 \end{pmatrix} \tag{3.1}$$

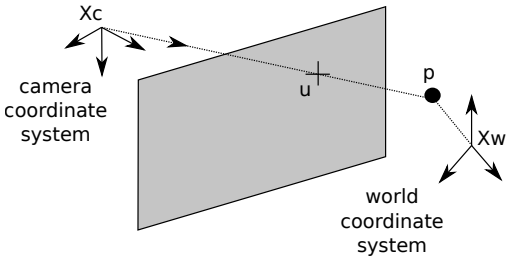


Figure 3.3: *Representation of the general pinhole camera.*

3.3.2 The General Pinhole Camera

Thus far we have assumed that our camera will have an ideal location at the origin of the of the world coordinate system; however, this is not a realistic assumption in practice. While it is true that every camera has a coordinate system attached to it, this almost always differs from the world coordinate system (Figure 3.3). Therefore, we must also consider this when building our mathematical model. A 3D point $\mathbf{p}$ can be described by both world coordinates $\mathbf{p_w} = (p_w^1, p_w^2, p_w^3)^t$ and camera coordinates $\mathbf{p_c} = (p_c^1, p_c^2, p_c^3)^t$ and usually $\mathbf{p_w} \not\equiv \mathbf{p_c}$. These two vectors are related by a rigid body transformation, specified by a translation vector $\mathbf{T} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$, such that

$$\mathbf{p_c} = \mathbf{R}\mathbf{p_w} + \mathbf{T} \tag{3.2}$$

The parameters $\mathbf{R}$ and $\mathbf{T}$ are called the *extrinsic parameters* of the camera. These values change every time the camera or projector is moved and therefore we need to re-calibrate them, as we will see more clearly in Chapter 5.

Using equation 3.1 and 3.2 we can describe every 3D point in relation to its 2D camera coordinates on the image plane:

$$\lambda\mathbf{u} = \mathbf{R}\mathbf{p_w} + \mathbf{T} \tag{3.3}$$

The remaining problems we must take into account in building our math-

ematical model are the following: the unit of measurement of lengths on the image plane (pixels) is not the same as that for world coordinates (meters), the distance from the center of projection to the image plane may be arbitrary, the origin of the image coordinates is usually in the upper left corner, the image plane may be tilted, the lens can distort the image, and the pixel may not be a perfect square.

These parameters for which we must compensate are called *intrinsic parameters* and we use a matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ to describe them. Our final equation is the following:

$$\lambda\mathbf{u} = \mathbf{K}\left(\mathbf{R}\mathbf{p}_\mathbf{w} + \mathbf{T}\right) \tag{3.4}$$

and the matrix $\mathbf{K}$ has the following form:

$$\mathbf{K} = \lambda \begin{pmatrix} fs_1 & fs_\theta & o^1 \\ 0 & fs_2 & o^2 \\ 0 & 0 & 1 \end{pmatrix}$$

where f is the *focal length*, s_1 and s_2 are used to compensate for non-square pixels, s_θ fixes the problem of tilted image planes and finally, o^1 and o^2 are the image coordinates of the intersection of the vertical line in camera coordinates with the image plane. It is very important to note that all the parameters of $\mathbf{K}$ are independent of the camera position. These can be calculated only once through calibration, because they describe physical properties related to the mechanical and optical design of the camera.

The following two Sections are very important. The first describes how to extract the parameters of every ray that extends from the *center of projection* through each pixel; the second one demonstrates how to recover the parameters of a projected plane that extends from the *center of projection* through each projected line.

Camera: Lines From Image Points

Each image point with coordinates $\mathbf{u} = (u^1, u^2, 1)^t$ defines a unique line $L = \{\mathbf{p_w} = \mathbf{q} + \lambda\mathbf{v} : \lambda \in \mathbb{R}\}$ containing this point and the center of projection. It is rather straightforward to obtain the parameters that describe the line L, because we know from equation 3.3 that $\lambda\mathbf{u} = \mathbf{R}\mathbf{p_w} + \mathbf{T}$, where $\mathbf{p_w}$ is a world point that is projected onto the image plane. Since $\mathbf{R}$ is a rotation matrix, we have $\mathbf{R}^{-1} = \mathbf{R}^t$. By rewriting the projection equation as

$$\mathbf{p_w} = (-\mathbf{R}^t\mathbf{T}) + \lambda(\mathbf{R}^t\mathbf{u})$$

we have just extracted all the parameters we need to describe our line L

$$L = \left\{\mathbf{p_w} = \mathbf{q} + \lambda\mathbf{v} = (-\mathbf{R}^t\mathbf{T}) + \lambda(\mathbf{R}^t\mathbf{u}) : \lambda \in \mathbb{R}\right\}$$

where $-\mathbf{R}^t\mathbf{T}$ is the center of projection.

Projector: Planes From Projected Lines

As previously shown, we can express the line in parametric and implicit form. The latter is very useful at this point:

$$L = \left\{\mathbf{u} : \mathbf{l}^t\mathbf{u} = \mathbf{l^1u^1} + \mathbf{l^2u^2} + \mathbf{l^3} = 0\right\}$$

where $\mathbf{l} = (\mathbf{l^1}, \mathbf{l^2}, \mathbf{l^3})^t$ with $\mathbf{l^1} \neq 0$ or $\mathbf{l^2} \neq 0$. As we will later see our projected pattern contains either horizontal or vertical lines. Therefore, the implicit equation of a vertical and horizontal line is

$$L_V = \left\{\mathbf{u} : \mathbf{l}^t\mathbf{u} = \mathbf{u^1} - \nu = 0,\ \mathbf{l} = (1, 0, -\nu)^t\right\}$$

$$L_H = \left\{\mathbf{u} : \mathbf{l}^t\mathbf{u} = \mathbf{u^2} - \nu = 0,\ \mathbf{l} = (0, 1, -\nu)^t\right\}$$

where ν is the first coordinate of a point on the line for L_V and the second coordinate of a point on the line for L_H.

There is a unique plane P that contains L and the center of projection. Again, from equation 3.3 we can extract the parameters to describe this plane P; we have

$$0 = \lambda \mathbf{l}^t \mathbf{u} = \mathbf{l}^t(\mathbf{R}\mathbf{p_w} + \mathbf{T}) = (\mathbf{R}^t\mathbf{l})^t(\mathbf{p_w} - (-\mathbf{R}^t\mathbf{T}))$$

On this basis, we understand that we can represent the plane P using its implicit form

$$P = \left\{\mathbf{p_w} : \mathbf{n}^t(\mathbf{p_w} - \mathbf{q}) = 0\right\}$$

where $n = \mathbf{R}^t\mathbf{l}$ and $q = -\mathbf{R}^t\mathbf{T}$.

Chapter 4

Human Body Scanner Setup

In this Chapter we will describe the setup we used to create our structured light 3D Scanner; recall that our main goal is to strive for simplicity and economy while maintaining the highest possible quality. Section 4.1 discusses how the different types of hardware can influence the quality of our scanner; Section 4.2 lists the software and libraries we have used in our project.

4.1 Hardware

We first had to choose what camera and projector would be the most suitable for our goals; indeed, there are many options each with its own advantages and disadvantages. After highlighting how the choice of camera and projector can affect the quality of the 3D reconstruction process, we will motivate the choices for the hardware chosen for the project.

4.1.1 Camera

This component is without a doubt the most fundamental part of the setup. A high quality camera grants a better analysis of the projected pattern, thus allowing for a better reconstruction of the 3D shape. Additionally, a good

camera is capable of detecting skin texture, which is essential in our project. However, a sophisticated piece of equipment can be very expensive; further it is important to balance the quality of the camera with that of the projector in order to avoid compatibility problems in the setup.

Summing up, the advantages that a high quality camera could add to the 3D scanner are the following:

- better texture quality
- better pattern reconstruction

Instead, the disadvantages are:

- higher resolution implies greater amount of time needed to send the image from camera to computer using only a USB connection
- higher resolution implies more computation and memory space needed to build the model
- can be very expensive

4.1.2 Projector

For our purpose, only two specifications for a projector are of importance. These are the brightness and the resolution.

The brightness factor is relevant for scans taken in a well-lit environment. If the projector's brightness is very low and the surrounding light very bright, the pattern projected onto a surface will be very difficult to detect. The brightness is determined by the number of *lumens* present, or the measure of the total amount of visible light emitted by a source.

The resolution factor is very important in building a highly detailed 3D model, because it allows for the detection of even the slightest changes in the scanned object.

4.1.3 Selected Hardware

It remains very important to obtain a high quality texture in the 3D model. That being said, in order to respect the goal of economy we will assume that the object will be scanned in a room that is very dark. This allows for the use of a projector with not so many lumens, i.e. a mid-range priced piece of equipment. This decision allowed us to use the best quality camera within the budget in order to detect even the smallest moles on the skin.

We chose to try two different projectors in conjunction with the camera. The first one is cheaper and with lower quality, the second one is a little more expensive but has more lumens and a higher resolution. The difference in using the former or the latter shown in Chapter 7.

Nikon D5000

(a) Front view (b) Side view (c) Top view

Figure 4.1: *Nikon D5000*

We chose to use the Nikon D5000 camera. Some important specifications are shown in Table 4.1:

Effective pixels	12.3 million
Image sensor	CMOS sensor, 23.6 x 15.8 mm
Image size (pixels)	4,288 x 2,848 [L], 3,216 x 2,136 [M], 2,144 x 1,424 [S]
Sensitivity	ISO 200 to 3200 in steps of 1/3 EV.
Exposure modes	Auto modes (auto, auto [flash off]), advanced scene modes (P), shutter-priority auto (S), aperture-priority auto (A), manual (M)
Interface	Hi-Speed USB
Dimensions (W x H x D)	Approx. 127 x 104 x 80 mm
Weight	Approx. 560g without battery, memory card

Table 4.1: *Nikon D5000 specifications.*

PicoPix 1430

(a) Front view (b) Side view

Figure 4.2: *PicoPix 1430*

The first projector is the PicoPix 1430, which is very compact and offers decent resolution. The specifications are shown in Table 4.2:

Display technology	VueG8* LCoS
Light source	RGB LED
LED light source lasts over	20,000 hours
Brightness	up to 30 lumens
Native resolution	800 x 600 pixels
Supported computer resolution	VGA (640x480, 60 Hz), SVGA (800x600, 60 Hz), XGA (1.024x768, 60 Hz), WXGA (1.280x768, 60 Hz)
Contrast ratio	500:1
Screen size (diagonal)	13.2 cm–205.7 cm
Screen distance	0.2 m–3.0 m
Focus	manual

Table 4.2: *PicoPix 1430 specifications.*

Philips cClear XG1 Brilliance

Figure 4.3: *Philips cClear XG1 Brilliance*

The second projector is the Philips cClear XG1 Brilliance, which has a higher resolution and much more lumens. The specifications are shown in Table 4.3:

Display technology	0.79" Polysilicon LCD x 3
Light source	RGB LED
LED light source lasts over	20,000 hours
Brightness	2600 ANSI Lumens (Normal)
Native resolution	1024 x 768 pixels
Supported computer resolution	XGA (1024 x 768), SXGA (1280 x 1024)
Contrast ratio	400:1
Screen size (diagonal)	76.2 cm - 769.62 cm
Focus	manual

Table 4.3: *Philips cClear XG1 Brilliance specifications.*

:

Finally, it is important to note that, given our choices, the scanner is compatible with any camera that can be controlled as a webcam or by the libgphoto2 library (Section 4.2.3), as well as with any projector that has a vga/hdmi connection. Therefore, it is possible to improve the quality of scans with updated hardware without difficulty, particularly considering the large volume of new cameras and projectors that come out every year with improved features and lower prices.

4.2 Software

In addition to substantial decisions made for the camera and projector, we made choices pertaining to which programming language we would use, what operating system would run our application, and therefore which IDE we would need to use to develop it. The following Sections will describe our choices.

4.2.1 Programming Language

The decision of which programming language to use to program the scanner was discussed at length. It was decided that portability would not be an important issue, whereas performance in both speed and memory would be paramount. On this basis, we chose C/C++ in order to take advantage of exhaustive libraries, as well as GPU processing using Nvidia CUDA or AMD FireStream for better image processing algorithms.

4.2.2 Development Environment

With regards to the operating system, the main choice was between Windows and a Linux distribution. We opted to use Ubuntu 10.04 LTS because it is free and does not require the purchase of a license to use it. Moreover, there are excellent libraries exclusive to Linux distributions, such as the *libgphoto2* (Section 4.2.3), which allow us to efficiently control the scanner's camera. For programming on Ubuntu we decided to use Eclipse CDT as the main IDE.

4.2.3 Additional Libraries

Some notable external libraries we used to build our application are the following.

libgphoto2 2.4.12

The libgphoto2 library [gPh] allows access to many digital cameras, through a C/C++ interface and a powerful API. With this capability it is possible to control the Nikon D5000 to take photos, which are then downloaded directly onto the computer's hard disk.

OpenCV 2.3.1

OpenCV (Open Source Computer Vision) [Opea] is a library written in C, with wrappers for C++, C# and Python. It provides programming functions for real time computer vision, it is cross-platform (Windows, Mac OS, and Linux) and it is free for use under the open source BSD license. It also has a C/C++ interface and more than 2500 optimized algorithms. We use it extensively for all the image computation necessary to build our 3D model.

OpenGL

OpenGL (for "Open Graphics Library") [Opeb] is a software interface to graphics hardware. The interface consists of a set of several hundred procedures and functions that allow a programmer to produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL is used to write the application that renders our 3D model and that allows to it to rotate and transform.

Chapter 5

Hardware Calibration

In Chapter 3 we explored the problem of understanding the mathematics needed to reconstruct our 3D human model. Although the concept of triangulation may sound deceptively easy, in practice we must pay great attention to all the matrices and vectors needed to solve the right equations.

In this Chapter, we will face one of the most important steps: the calibration of our camera and projector. In order for triangulation to work, we need to calculate the values of λ, $\mathbf{K}$, $\mathbf{R}$, $\mathbf{T}$ of equation 3.4. This formula can be applied to both the camera and the projector. The only distinction being the direction in which light travels: for the camera, light travels from the world towards its image plane, whereas the projector works as a sort of an inverse camera, wherein the light is generated from the center of projection and radiates onto the world. Therefore, we discuss in Section 5.1 two slightly different calibration procedures, which take this difference into account, by calibrating both the camera and the projector separately. Finally, Section 5.2 shows the calibration results for the hardware shown in Chapter 4.

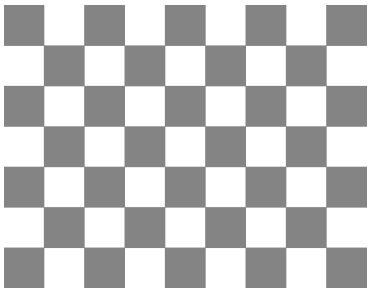


Figure 5.1: *The chessboard pattern used in our calibration procedure.*

5.1 Camera and Projector Calibration

We present a very simple calibration procedure using a printed black and white chessboard pattern based on the established method of Zhang [Zha99] [Zha00]. Using his method, the parameters of the general pinhole camera model will be recovered.

Today, camera calibration tools are very common, well documented and freely available; however, tools for projector calibration are not so easy to find. We present here our approach based on the excellent work of [LT09]. We begin by describing the technique proposed by Zhang. We finally discuss our own implementation and provide step-by-step directions on how to calibrate the camera and projector using the software we developed.

5.1.1 Calibration Methods

There are many different possibilities when estimating the parameters of the general pinhole model. Some are connected with particular camera models that are used and work well in conjunction, while others are more general and more easily adaptable to our goals. Regardless, the main goal of all calibration procedures is ultimately to evaluate the *intrinsic parameters* (focal length, principal point and scale factor) and the *extrinsic parameters* (rotation matrix and translation vector) of the camera. For a better understanding of camera

models and calibration methods see [FCWC08] and [HZ04].

The general way to overcome the burden of calibration is by taking pictures of a sequence of calibration objects with clearly distinguishable features. The correspondences of these unique features in the various images provide a set of 2D to 3D points, which are then used to evaluate the camera and projector parameters. As previously mentioned, many different techniques may be used; in most community-developed tools, the most widely adopted method is that originally proposed by Zhang. This method requires the use of a planar chessboard pattern (Figure 5.1) observed in at least two different positions; indeed, more positions provide a more accurate evaluation of the parameters. From this sequence of pictures, the intrinsic parameters can be extracted using a factorized approach. From this point on, the extrinsic parameters can be calculated by taking only one picture. This is of great importance considering that the camera and projector we use in our project have been moved around a lot in order to test different scenarios and improve the quality of our scanner. Therefore, a method that allows us to quickly and efficiently recompute the extrinsic parameters is fundamental. Last but not least, the Zhang method is so commonly used because it requires only a printed pattern, rather than a particular calibration object, to calibrate the camera.

5.1.2 Calibration Software

A variety of software can be found and used, sometimes for free, to calibrate the camera. The most powerful and reliable is without a doubt MATLAB [Mat], which has an Image Acquisition Toolbox [IAT] that supports products from a variety of vendors, as well as any DCAM-compatible FireWire camera or webcam with a Window Driver Model (WDM) or Video For Windows (VFW) driver. Despite this widespread availability, we require the inclusion of a calibration procedure within our program to avoid the need to rely on any additional software. We have therefore chosen OpenCV, instead of MATLAB.

OpenCV (see Section 4.2.3) provides all the necessary functions to build our optimized calibration procedure, and much more.

5.1.3 Calibration Procedure

Our goal for the calibration procedure was to maintain the highest possible level of simplicity. As previously mentioned, the Zhang method requires us to take pictures of a printed chessboard pattern in multiple positions. Then, the corners of the printed chessboard pattern are detected in all the images and linked to the same corners in the previous pictures. Finally, an equations system is solved and the parameters are known. This procedure is the standard for camera calibration; however, for projector calibration we have to use a different approach: instead of analyzing a printed chessboard, we must analyze a chessboard pattern that is projected on top of a printed one. The printed and projected patterns must have the same structure (i.e. same number of rows and columns). This modification is very minimal and it allows us to reuse much of the camera calibration code, as well as to calibrate the camera and projector together by taking an initial picture of the printed pattern and then a second immediately afterwards of the same printed pattern beneath a projected one.

In order to achieve a timely calibration procedure, it becomes immediately clear that we must solve three problems automatically: detect the corners of the printed pattern, link them together (top left corner in picture 1 should be associated to top left corner in picture 2, 3, etc.), and extract the parameters from the equations.

We will explain how we solve these issues step-by-step, but first we go back to step 1: taking the pictures of the printed pattern.

This is the easiest problem; using libgphoto2 (Section 4.2.3) we are able to take any picture we desire and save it directly onto our hard disk. Therefore, in step 1 we have to hold our printed pattern in front of the camera and

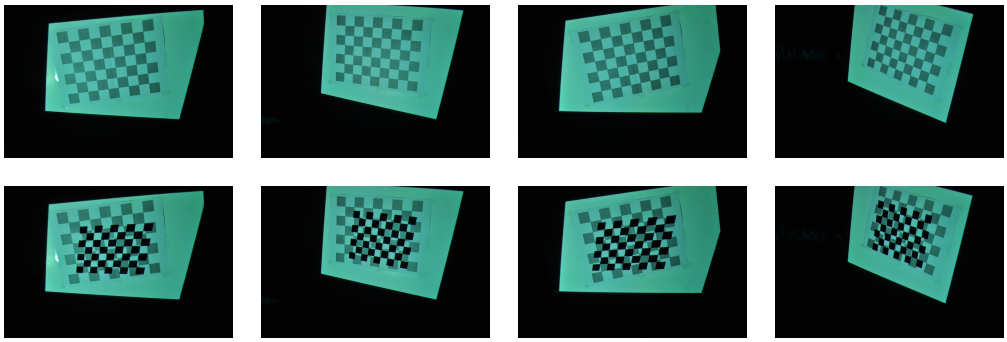


Figure 5.2: *Calibration sequence containing multiple views of a printed chessboard pattern on the first row, and a printed chessboard pattern plus the projected pattern on the second row.*

the projector simultaneously. We begin by taking a picture of the printed pattern and then, a few milliseconds later, of the projected pattern on top of the printed one. Next, we start to analyze these two initial pictures, thus providing the user time to slightly move and tilt the printed pattern. We then take two more pictures, and continue this way until we have enough pictures to extract the intrinsic and extrinsic parameters of camera and projector. The printed pattern must be presented in at least two different positions, requiring a total of four pictures; however, the more positions in which we can analyze the printed pattern the more accurate and robust our calibration will be. We found that between six to ten different positions allow for an acceptable compromise of time vs. accuracy. See figure 5.2 as an example of pictures taken of the printed and projected pattern in a variety of orientations.

After every two pictures taken, that is, a printed pattern picture followed by a printed pattern plus projected pattern picture, we perform an analysis in order to automatically detect the corners of the pattern in each image. Luckily, OpenCV gives us two useful built-in algorithms to solve this problem. The first one is called *cvFindChessboardCorners* and returns a pointer to the structs of the 2D points in each image, corresponding to a detected corner.

The second is called *cvFindCornerSubPix* and from the previous array of structs finds a subpixel approximation of the corners by performing a local search around each of them. This guarantees a better quality of calibration results; see figure 5.3 for an example.

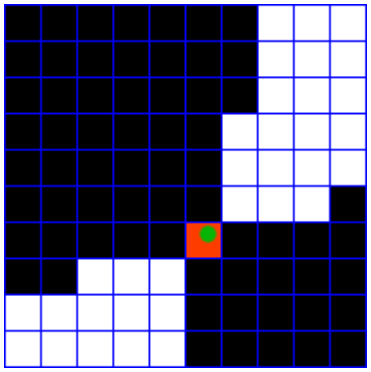


Figure 5.3: *Example of subpixel accuracy on corners. The orange square is the detected corner from cvFindChessboardCorners, the green point is the refined 2D coordinates found by cvFindCornerSubPix.*

At this point, we need to determine whether we are able to detect all the internal corners of the pattern or not. If we are, then we can move on to the last step, otherwise we need to take more pictures of the pattern.

The final step is to use the information we have gathered thus far about corner location to extract the camera and projector parameters. This is done by solving the Zhang equations. For camera calibration, it is sufficient to leverage the power of OpenCV again. Calling *cvCalibrateCamera2* on the array storing the corner locations, we get in return a pointer to the intrinsic camera parameters matrix. For calibrating the projector, however, we require some additional steps. Firstly, we must evaluate the undistorted image pixels for both the camera and the projector chessboard corners. Then we estimate the homography (an invertible transformation from a projective space to itself that maps straight lines to straight lines) that maps the undistorted image pixels to their positions on the chessboard. Next, we map the undistorted

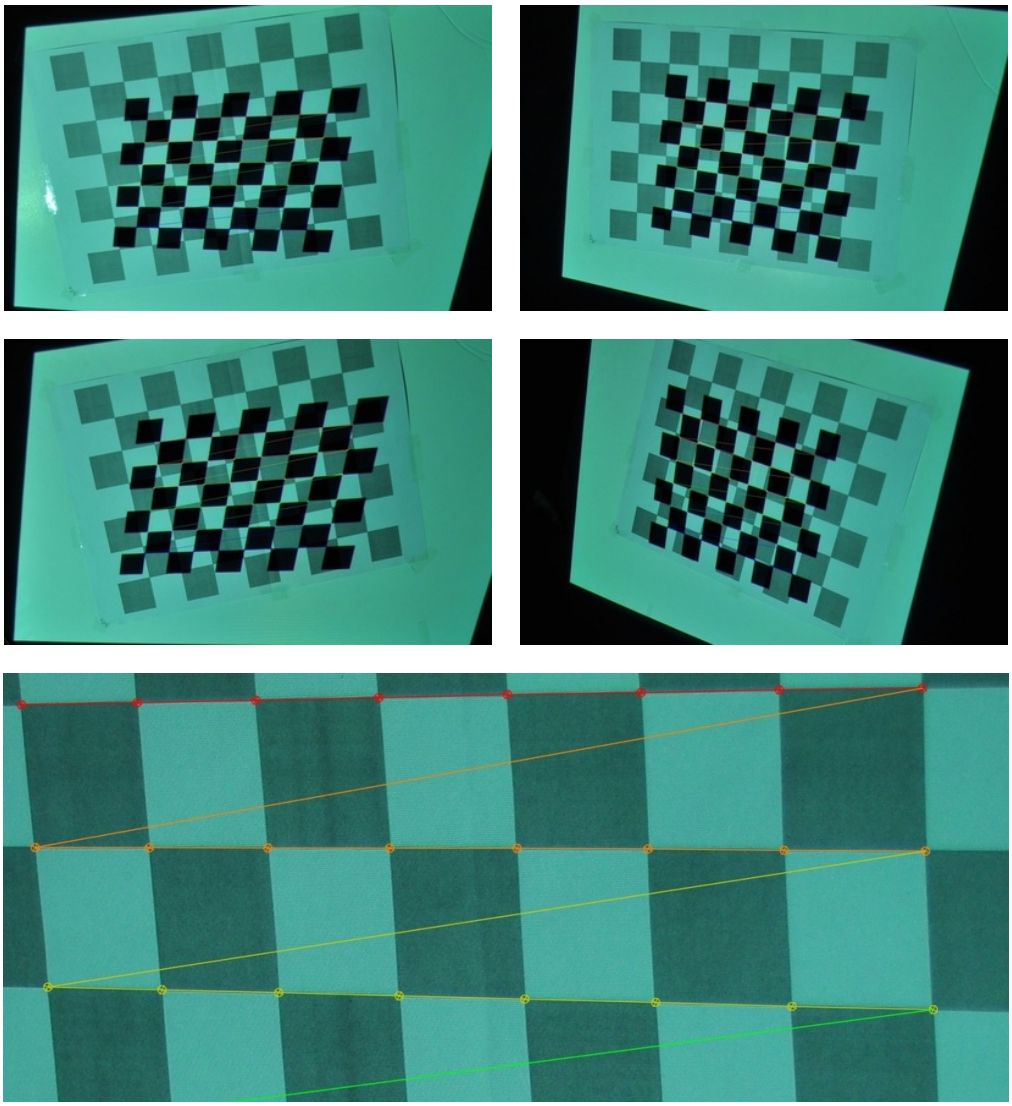


Figure 5.4: *Detected corners on the calibration sequence with projected patterns.*

projector image corners to the undistorted chessboard corners. Finally, we are able to define a few object points that express the relationship between the projector chessboard corners and the corresponding plain chessboard pattern corner. Now, we use *cvCalibrateCamera2* with the object points we have just created to extract the intrinsic projector parameters.

Last but not least, we can use the function *cvFindExtrinsicCameraParams2*

and the first calibration image to extract the extrinsic parameters that define the position of the camera and projector in relation to world coordinates.

5.2 Calibration Results

We will discuss in this Section the effectiveness of our calibration procedure in terms of time, accuracy and effort.

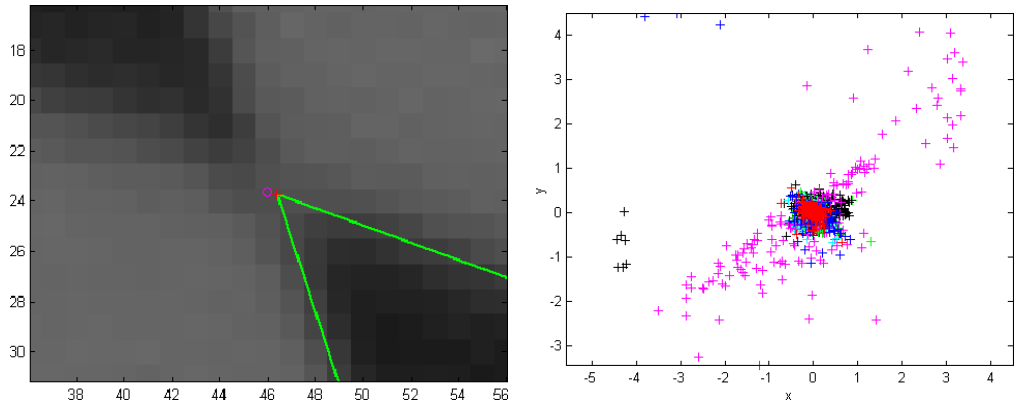
Time We divide the time needed to calibrate the camera and projector into two parts: time needed to acquire the images and time needed to analyze images and recover the intrinsic and extrinsic parameters.

The first is a very aleatory value that depends on the simple fact that we sometimes need to retake two pictures, as in the instance that we were not able to detect the right amount of corners in the initial images. We repeat this process until all the corners are detected. Occasionally, the environment light combined with the tilting of the checkerboard pattern is problematic and interferes with the software's detection of the corners. Generally, however, the user is able to facilitate the recognition of the pattern straight away, particularly after gaining some manual experience with the calibration procedure. The time needed to take two pictures is about one second.

The last parameter that may influence the time needed to calibrate our setup is the processing time. We found that the only function that really takes a significant amount of time is *cvFindChessboardCorners* , which can take anywhere from less than one to up to thirty seconds to detect all the corners. This is primarily due to the fact that the pictures have a resolution of 4288×2848 and the function iterates over every pixel, thus spending a decent amount of time on each of them. All other processing work can be completed very quickly.

Generally, the time required to calibrate the camera and the projector with seven different positions of the pattern is around three minutes. Afterward, the extrinsic calibration of our setup can be done by taking just one picture in less than fifteen seconds.

Accuracy While time elapsed can be an interesting way to measure the quality of our calibration procedure, the most important factor by far is accuracy. The best way to test the precision of our extracted parameters is to reproject the corners on top of the previous images and determine how much they differ from the original. We do not implement this part of the process in our program; however, by using MATLAB [Mat] and its Image Acquisition Toolbox [IAT] we are able to do so seamlessly, as we see in figure 5.5. The results are exceptionally favorable and guarantee that our calibration procedure is robust and efficient.



(a) Reprojected Corner

(b) Reprojected Errors Plot

Figure 5.5: *(a) Reprojected corner (circle) and original one (+). (b) a plot with all the errors in pixel.*

Effort The final factor worth mention in our calibration results is the level of difficulty for a user who goes through the process and calibrates the setup. We have seen that after an initial adjustment with the calibration procedure, anyone can easily calibrate the camera and projector, even without the knowledge of the inner workings of our program.

Chapter 6

3D Surface Modeling

In this Chapter we describe how our structured light scanner works. In particular, we focus on: the choice of pattern to be projected (Section 6.1), the capture of the images of the patient with the superimposed pattern (Section 6.2), the analysis of said images (Section 6.3) and, finally, the reconstruction of the 3D model by triangulation (Section 6.4). Section 6.6 of this Chapter explains how we use OpenGL to build a program that generates the 3D model. From this point onwards, we assume that both the camera and the projector are calibrated and we know the intrinsic and extrinsic parameters for each of them.

6.1 Projected Pattern

For future references, we assume that our projector works with a resolution of 1024×768. Different projectors may have different resolutions and our program adapts easily to this change, but here we will use the aforementioned values for the projector's resolution to simplify the explanation of our program.

The primary benefit of using a projector rather than a laser-beam is to remove the mechanical movement of the latter, required to sweep the beam of light across the real-world surface to be reconstructed. Additionally, a

projector is capable of displaying any arbitrary color image and, therefore, provides a new and interesting challenge in relation to our scanner: which pattern is best to project onto our patient in order to reconstruct a body in the most accurate way?

The simplest method would be to use the projector as a laser-beam by projecting a single column/row of white pixels translating against a black background. Thus, we would need to capture exactly 1024/768 pictures. Afterwards, the model point cloud is reconstructed using familiar ray-plane triangulation, explained in Chapter 3. In doing so, we immediately see that we are not utilizing the full power of the projector by ignoring its capability to display arbitrary color images. We would like to use less frames for the model reconstruction, making the scanning process much faster while maintaining the necessary accuracy. The only way to improve the speed of the scanner is to display a more complex pattern than a single column/row of white pixels. This is done in two different ways: encoding the projected planes *spatially* (i.e., within a single frame) or *temporally* (i.e., across multiple frames: the simpler projected pattern idea discussed above can be understood as a very inefficient temporally encoded pattern). There are both benefits and drawbacks to either strategy. For instance, purely spatial encodings allow for the use of a single pattern, enabling fast reconstruction of even dynamic scenes. Alternatively, purely temporal encodings are more likely to benefit from redundancy, consequently reducing reconstruction artifacts. Finally, a combination of spatial and temporal encodings may be used. For an overview of the most common techniques used today, complete with an accurate analysis on the quality of each, we suggest the reader review [SPB04].

Following an in-depth analysis of possible patterns we might use, we decided to focus on a temporally encoded pattern. We see that with the pattern to be be discussed next we are able to reconstruct a very good 3D model using $\sim$20 pictures; however, we remain interested in how well our

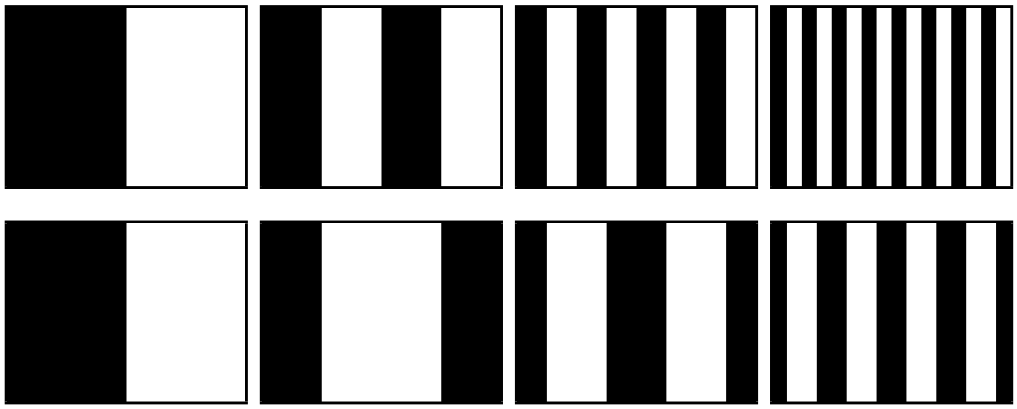


Figure 6.1: *Structured light illumination sequences.*
(Top row, left to right) The first four bit planes of a binary encoding of the projector columns, ordered from most to least significant bit.
(Bottom row, left to right) The first four bit planes of a gray code sequence encoding the projector columns [LT09].

scanner performs using only one spatially encoded pattern. We refer to Chapter 8 for more information about this.

A very simple, yet powerful, temporally encoded pattern is a binary structured light sequence. Our projected pattern is a gray code structured light sequence and is an improvement on a binary structured light sequence. First, we will describe the binary codes theory in order to better understand its potentiality, and consequently the even greater effectiveness of gray codes. The simple binary structured light sequence was first introduced in [PA82]. As shown in figure 6.1, the projection consists of a sequence of frames that are divided into white or black columns[1]. Each projected column defines a single bit in the binary representation for the column. For example, column 115 has a binary representation of 1110011, which implies that in the first three frames it will project a white column, in the 4th and 5th frames it will project

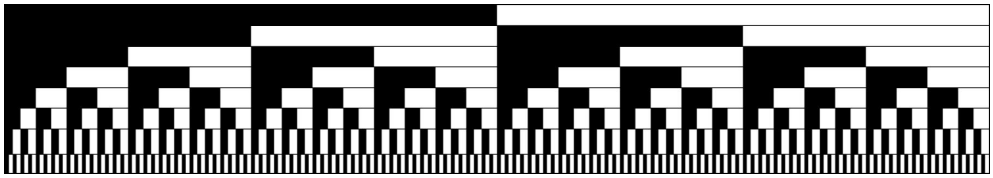
[1]For the sake of explanation, we also assume that we are projecting a column of white or black pixels. If we had to project rows, the same concept still applies.

a dark column and in the last two frames it will project a white column. In this way, by analyzing the captured images and applying a threshold to them, it is possible to decide whether each pixel is white or black, thus associating the projected plane with each pixel. The ray-plane triangulation can finally reconstruct the 3D model.

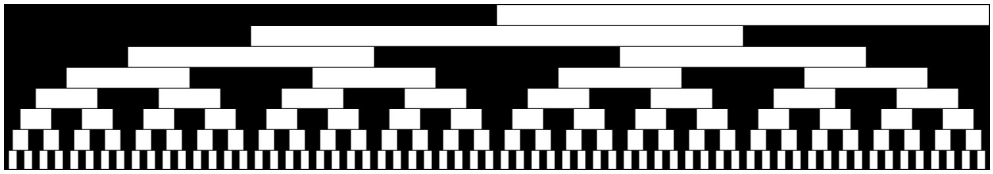
We quickly see the power of a binary structured light sequence. Using this sequence, it is possible to simplify the portion of the program that is concerned with image analysis. In fact, using a single threshold for each captured picture, it is possible to immediately reconstruct the 3D model. Why do we need to improve our binary codes and opt for using gray codes instead? It has been proven that binary codes are not robust with channel noise; therefore, errors in the assignment of a pixel value (black or white) can lead to large reconstruction errors. Gray codes were first proposed as an alternative to simple binary encoding in [ISM84], and have since been proven to be a stronger alternative to binary codes, as in the detailed analysis in [SPB04].

A gray code structured light sequence can be created from a binary sequence, and vice versa (Figure 6.2). The key property of gray codes is that two neighboring code words (for example column 15 and 16) only differ by one bit. As a result, an error in the assignment of a pixel value will merely lead to the intersection of the ray going through that pixel with a plane that is off by just one column. The error in the 3D model will be very minimal.

The last element worthy of mention is the number of pictures we must capture using a gray code structured light sequence. This number depends on the resolution of the projector. Assuming the projector has a resolution of 1024×768 and we are projecting column gray code patterns, then we need to capture $2\lceil \log_2 1024 \rceil + 2$ different pictures. The total number of projected patterns is $\lceil \log_2 1024 \rceil$, and for each of them we need to project the inverse pattern to simplify the reconstruction process later on (see Section



(a) Binary structured light sequence



(b) Gray code structured light sequence

Figure 6.2: *Comparison of binary (top) and gray code (bottom) structured light sequences. Each image represents the sequence of bit planes displayed during data acquisition. Image rows correspond to the bit planes encoding the projector columns ordered from the most to the least significant bit (from top to bottom) [LT09].*

6.3). Moreover, we need to acquire a picture of the patient by projecting a completely white image and another by projecting a completely black image. The former is used to recover information about the texture; the latter is XORed with the first. The resulting image is used as a mask for what we should or should not consider in the following images.

6.2 Image Capture

Temporally encoded patterns, like our gray code structured light sequence, require the subject to remain completely still to allow for better reconstruction of the model. We continue to study the best way to photograph our patient to recover the 3D model; an insight into this aspect of our research can be read in Chapter 8. At this step in our process to recover the model of the patient,

we are primarily concerned with reconstructing single parts of the body and attaching texture information to their 3D models. To better demonstrate the reconstruction procedure, we will use an elucidative example: we will attempt a reconstruction of the shape of two hands.

The way we capture the pictures is very straightforward: we place the subject in front of the camera and projector and then take a picture every time the projected pattern changes. The devices are synchronized by our program and the acquisition time of 22 pictures is about 20 seconds. Some of the captured frames are shown in Figure 6.3, where a gray code horizontal pattern is projected.

6.3 Image Analysis

After the calibration procedure, image analysis is the most delicate, yet important part of our program. The quality of our 3D model greatly depends on the quality of our analysis. In fact, the analysis of each captured image decodes the structured light sequence and assigns a projected plane to each pixel. The process is actually very straightforward, and will be described using pictures to clarify a few steps. In general, we must determine whether the projector directly illuminates each pixel in each displayed image. If it does, then the corresponding code bit of that pixel in that frame is set to 1; otherwise, it is set to 0. Knowing these values for each frame allows us to understand which code belongs to the plane projected on top of that pixel. For example, if we are analyzing a pixel at coordinates (100, 100) and we know that it is directly illuminated by the projector in only the first and last frame, then we know that it is illuminated by plane 1000000001 alias plane 513 in decimal.

We need to find a robust way to determine whether or not a pixel is illuminated by the projector. It has been proven that applying a single fixed

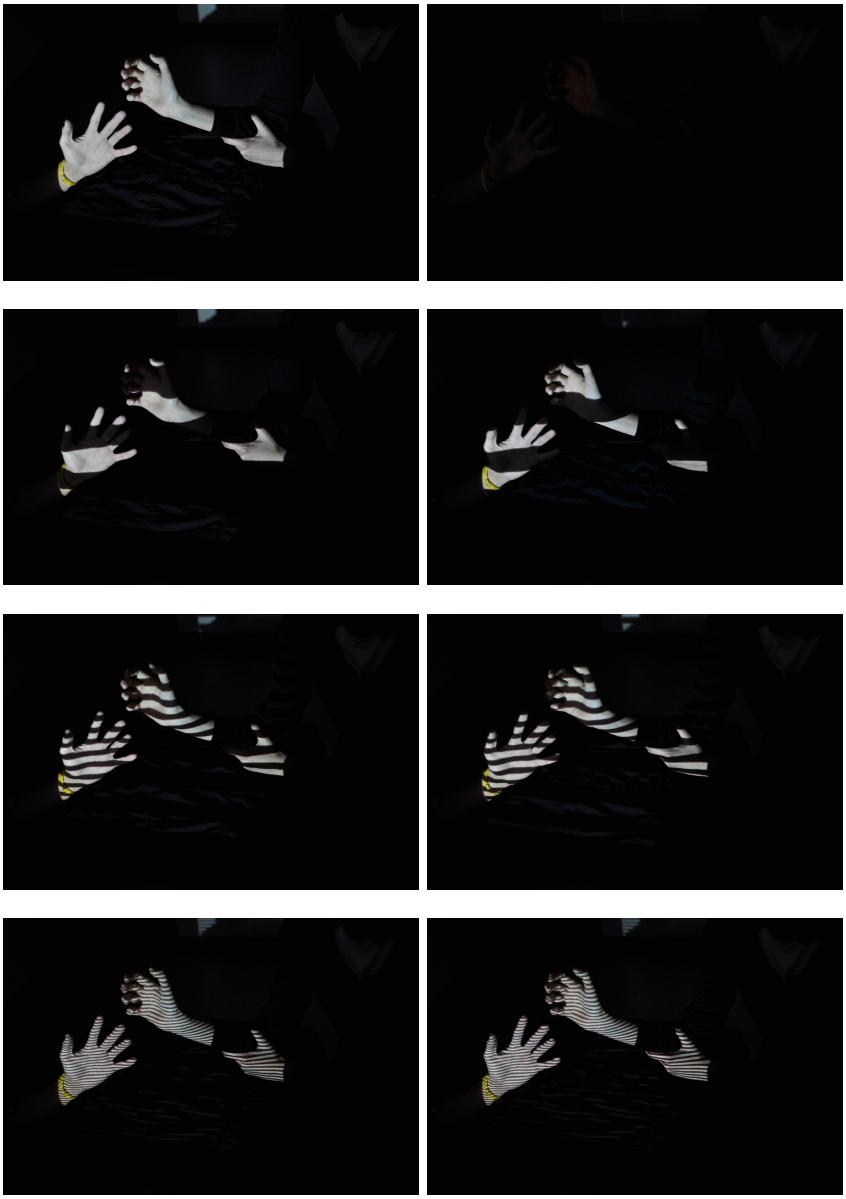


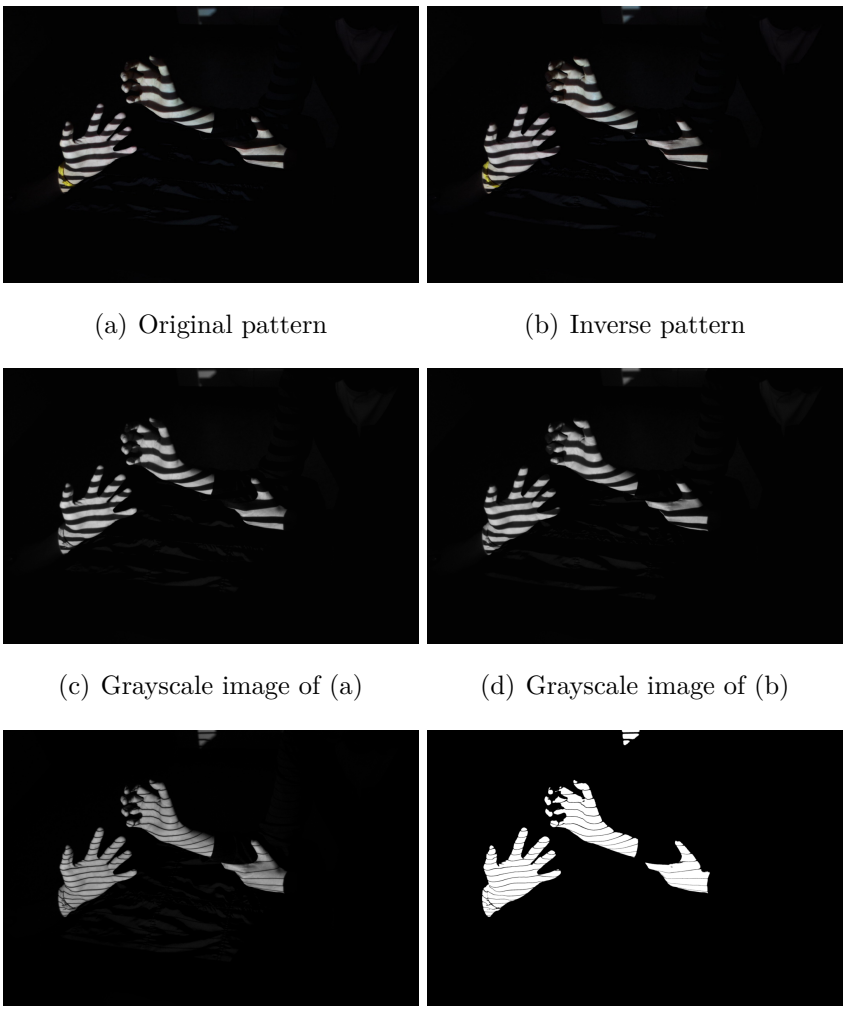
Figure 6.3: *Some frames of the captured sequence. The first column shows the result of projecting an all white image plus the original gray code pattern, whereas the second column displays the results obtained by projecting the inverse pattern. It is then used for an easier recovery of the 3D model, as explained in Section 6.3.*

threshold to the image results in decoding artifacts. For instance, certain points on the surface may only receive indirect illumination from directly illuminated points. This may cause a bit error, in which an unilluminated point appears illuminated due to scattered light. Such bit errors may produce significant reconstruction errors. One proposed solution is to project an inverse pattern in addition to the original pattern. While we need approximately double the amount of captured images, the decoding process is less sensitive to scattered light because a variable per-pixel threshold may be used. A pixel is now determined to be illuminated or not, depending on whether the projected pattern or its inverse is brighter. Figure 6.4 shows the subject illuminated by an all-white image, the decoded row indices and the decoded depth map for each pixel. The decoded row indices are colored from green (row index number 767) to blue (row index 0); in quite the same way, the depth map is colored from green (closer points to the camera) to blue (points which are farther away from the camera).

(a) All-white image (b) Decoded rows (c) Depth map

Figure 6.4: *Decoding the gray code structured light sequence. Note the shadow on the fingers of the right hand, prohibiting the reconstruction of parts of the fingers.*

Now, we will look at one pair of frames and analyze the steps taken to decode the row indices for each pixel. The results are illustrated in Figure 6.5 and Figure 6.6.

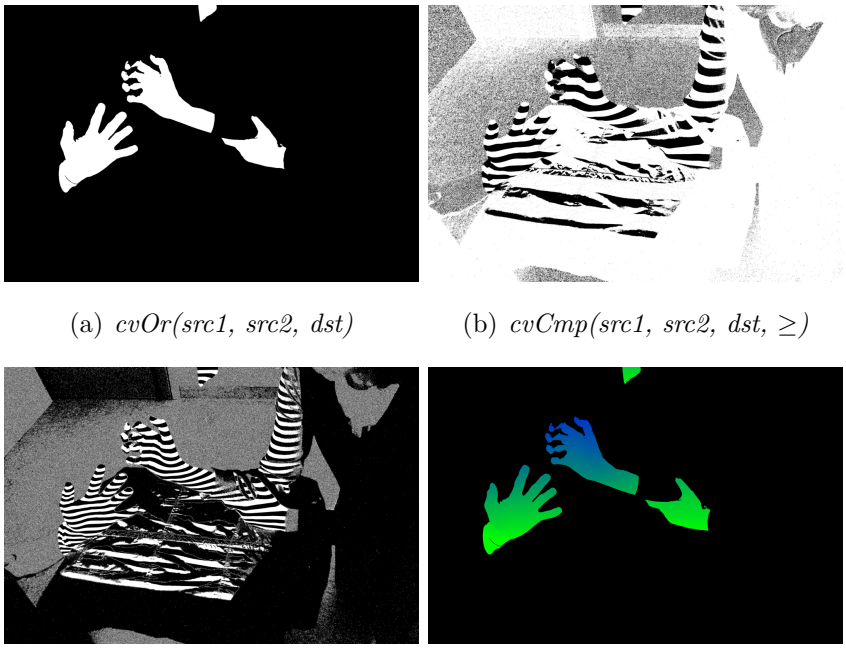


(a) Original pattern (b) Inverse pattern

(c) Grayscale image of (a) (d) Grayscale image of (b)

(e) *cvAbsDiff(src1, src2, dst)* (f) *cvCmpS(src, threshold, dst, ≥)*

Figure 6.5: *(a), (b) captured images of the 5th projected pattern. (c), (d) per-pixel threshold to convert image (a) and (b) to grayscale. (e) absolute difference between image (c) and (d) such that* $\mathtt{e}(i) = |\mathtt{c}(i) - \mathtt{d}(i)|\ \forall i \in$ *pixel. (f) fixed threshold to (e).*



(a) *cvOr(src1, src2, dst)* (b) *cvCmp(src1, src2, dst, ≥)*

(c) B&W decoded row indices (d) Colored decoded row indices

Figure 6.6: *(a) OR with the masks calculated in all the previous frames. (b) assignment of a high or low bit to each pixel, depending on whether the pixel is brighter in picture 6.5(c) or 6.5(d). (c) adding the decoded row indices of (b) to the previous calculated one. (d) colored version of (c) to better display the decoded row indices; the colors range from green (row index number 767) to blue (row index 0).*

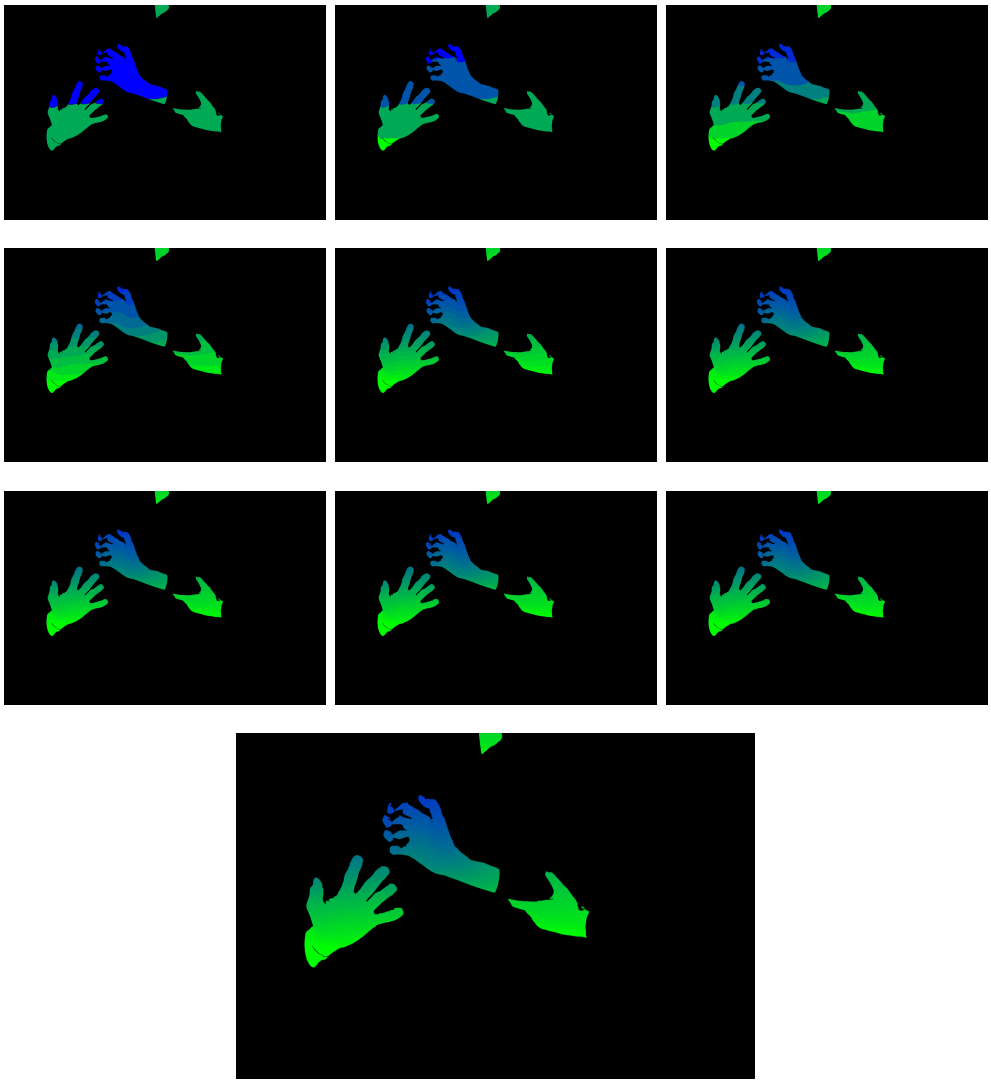


Figure 6.7: *Sequence of the decoded row indices. The last bigger picture is the final step.*

We now understand how to process each pair of images in order to decode the row indices. We end this Section by showing how much precision we add in every step to decode the row indices, as in Figure 6.7 . For this purpose, we show the colored decoded row indices for each pair of images, where the improvements after each step become immediately apparent.

6.4 Image Triangulation

Once we know the correspondence of the decoded row indices to the camera pixels, the reconstruction of a 3D colored point cloud is made rather straightforward by ray-plane triangulation. Ultimately, a simple per-point RGB value can be assigned to each 3D point by sampling the color of the all-white picture (see Figure 6.4(a)).

6.5 3D Model

At this point, the majority of the task is complete; however, we have only a 3D colored point cloud of our subject. The next Sections will describe how we convert this point cloud into a surface, as well as the best way to save these results to a file, storing all this information.

6.5.1 Surfaces from Point Clouds

The creation of watertight surfaces is fundamental in the construction of a 3D model. This allows a model with a surface to be rendered with higher quality, allowing for a more accurate analysis of the model itself. In general, there are two main surfaces that can be fitted using our point cloud: *discrete surfaces* and *isosurfaces*. An isosurface is a mesh that approximates a smooth implicit surface $S = (x : f(x) = 0)$; it is a powerful way to describe our model, but its calculation requires complex algorithms that we will not consider in this project. We have chosen to create a simple discrete surface that still fulfills all our requirements.

The general task of using a 3D point cloud to compute a discrete surface consisting of many connected triangular meshes can be greatly simplified here, since we are able to exploit preexisting ordering constraints of the regular pixel grid. Each reconstructed 3D point belongs to one of the 768 projected planes;

we can order every 3D point according to the projected plane it belongs to and, finally, connect every point with the point to the right and the point belonging to the line below. One must take care when ordering points in each triangular mesh to be sure that all the meshes have their normal vectors pointing into the same direction (see Figure 6.8).

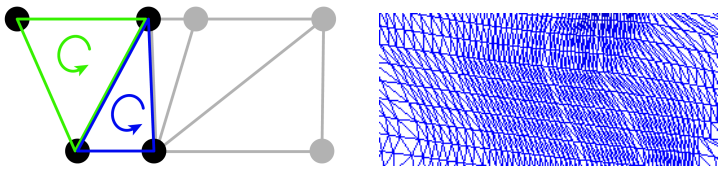


Figure 6.8: *Creation of a watertight wireframe model [FER07].*

The last step in building the watertight wireframe model is a cleanup process. Many existing meshes are wrong because of border errors, as well as other errors, which occur at light absorption areas (such as hair and eyes) or on very shiny areas; therefore, we apply a triangle area filter. This filter evaluates the triangle areas and removes those triangles that do not meet the defined specifications, e.g. size. Finally, we recover the texture information from the all-white image of Figure 6.4(a) and assign it to that mesh.

6.5.2 Saving the Model as a VRML File

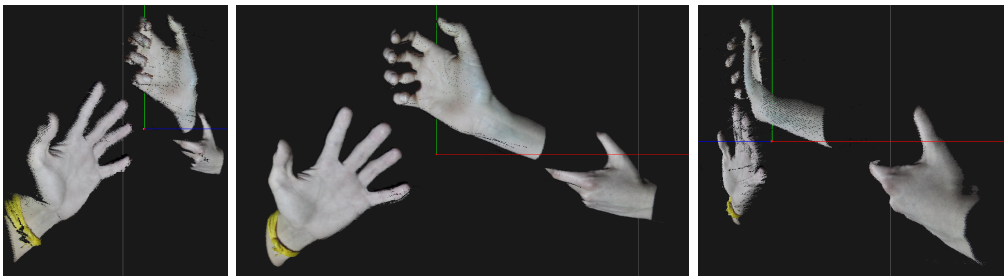
Storing and retrieving our 3D model is fundamental, and saving it in either ASCII format or binary format is a valid solution to the problem. We have chosen to use the Virtual Reality Modeling Language (VRML) to store our model [VRM]. VRML is an ISO standard published in 1997 that describes in ASCII format a scene graph comprising of different nodes. The geometry node that we use is called *IndexedFaceSet*. It is designed to store point clouds with their 3D coordinates, colors and meshes. A simple VRML file looks like that in Figure 6.9.

```
#VRML V2.0 utf8
Shape {
  geometry IndexedFaceSet {
    coord Coordinate {
      point [  -162.507828     189.541809     1034.713379
               -162.244995     189.527145     1034.636719
               -163.760605     189.297012     1034.709229
               -163.234772     189.267746     1034.555786
      ]
    }
    coordIndex [  0 1 2 -1,
                  1 3 2 -1,
                  3 2 1 -1
    ]
    color Color {
      color [  0.152941     0.145098     0.149020
               0.164706     0.156863     0.160784
               0.094118     0.086275     0.090196
               0.145098     0.137255     0.141176
      ]
}}}
```

Figure 6.9: *Example of a VRML file.*

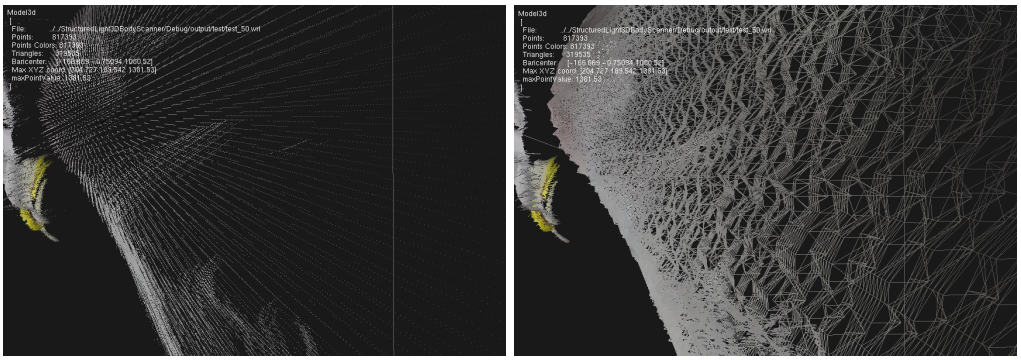
6.6 Visualization with OpenGL

A well-established technique to render point clouds and triangular meshes uses OpenGL. OpenGL simplifies the display of points and associated polygons, such as triangles or quads. Moreover, its powerful API takes advantage of CPUs and GPUs, making the rendering of the 3D model fast and smooth. We developed a program that reads the 3D model VRML file and displays it on a screen, and has also been optimized to display up to 10 million points fluidly. Some shortcuts have been added to easily switch among a point cloud, a wireframe and a texture visualization. In addition, it is possible to move the model simply by using the mouse buttons and the keyboard WASD keys. The Figures 6.10 and 6.11 show snapshots of the rendering program displaying the reconstructed hands and arm.

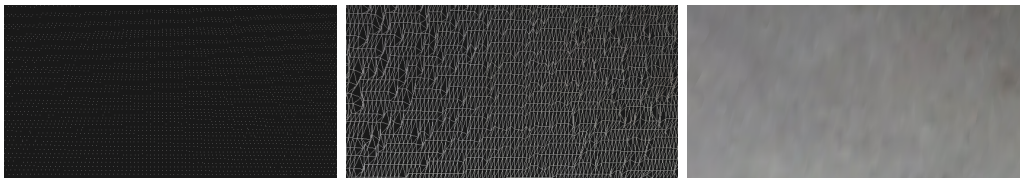


(a) Left side view (b) Front view (c) Right side view

Figure 6.10: *Visualization of the 3D colored point cloud. It is possible to see some dark dots in the hand that corresponds to pixel that have not been reconstructed (the causes can be many and are better discussed in the following Chapter).*



(a) Point cloud of the arm (b) Wireframe of the arm



(c) Point cloud (d) Wireframe (e) Meshes

Figure 6.11: *A zoom on the arm. Subfigures (a) and (b) display a general view of the arm. (c), (d) and (e) show the detailed difference among a point cloud, wireframe and meshes visualization.*

Chapter 7

Performance

This Chapter discusses the quality of the 3D models we create with our human body scanner. In the following Sections, we compare results obtained using two different projectors and realize the great importance a projector has in determining the quality of a model. Finally, in the last Section we compare our scanner with a professional laser scanner. We recall that our goals are: accuracy, efficiency and costs.

7.1 Accuracy

The scanner must be very precise and able to detect body changes of less than one millimeter. To study how well our scanner performs, we have built the test object shown in Figure 7.1. The test object is made up of four

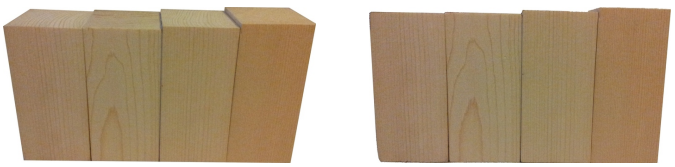
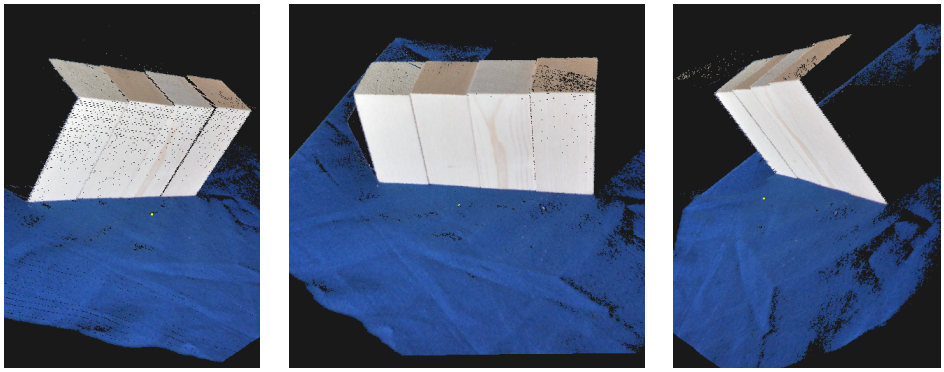


Figure 7.1: *Test cubes used to measure the accuracy of our system; note the different heights of the four parallelepipeds.*

parallelepipeds with four different heights: 10 cm, 10.1 cm, 10.3 cm, and 10.6 cm. When they are placed close together, they resemble a staircase. Scanning this object allows us to better understand if we are able to detect even the slightest changes on the surface. Our test goal is to detect the smallest step, a difference of 1 millimeter.

In the following paragraphs we discuss the accuracy of our system using a PicoPix 1430 versus a Philips cClear XG1 Brilliance.

PicoPix 1430 This projector was chosen for its compact size and decent resolution of 800×600; however, it is severely lacking in brightness, with only 50 lumens at its disposal. Figure 7.2 shows the 3D model of the test cubes that we created using the PicoPix projector. Even a projector with very few



(a) Left side (b) Front side (c) Right side

Figure 7.2: *The 3D model of the test subject, reconstructed using the PicoPix 1430.*

lumens is able to reconstruct a good 3D model. In fact, we see in Figure 7.3 that our scanner is able to detect even the 1 millimeter change on the surface of the test cubes.

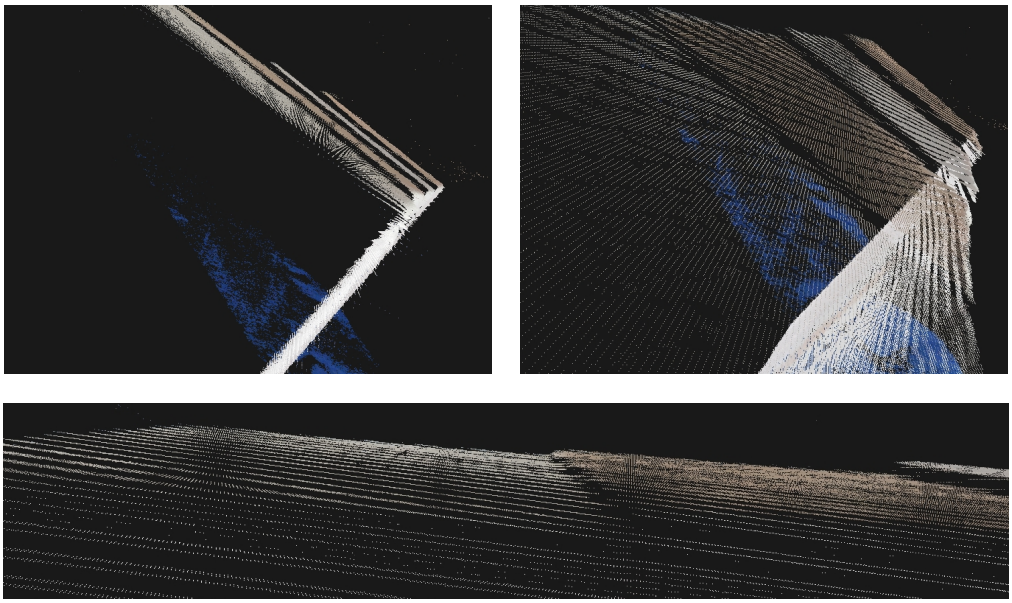
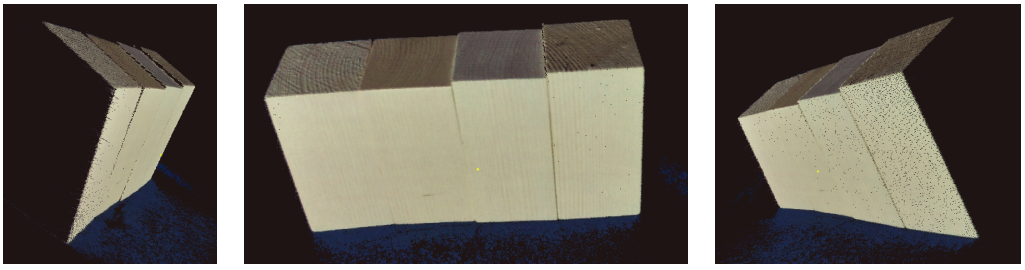


Figure 7.3: *Zoom on the 3D model to prove that we are able to detect even the 1 mm change on the surface of the test cubes.*

Philips cClear XG1 Brilliance The Philips cClear XG1 Brilliance is a more powerful projector compared to the PicoPix 1430. It has a higher resolution of 1024 × 768and 2600 lumens to use. On the other hand, it is much bigger, noisier and costs a little more.



(a) Left side (b) Front side (c) Right side

Figure 7.4: *The 3D model of the test subject, reconstructed using the Philips cClear XG1 Brilliance.*

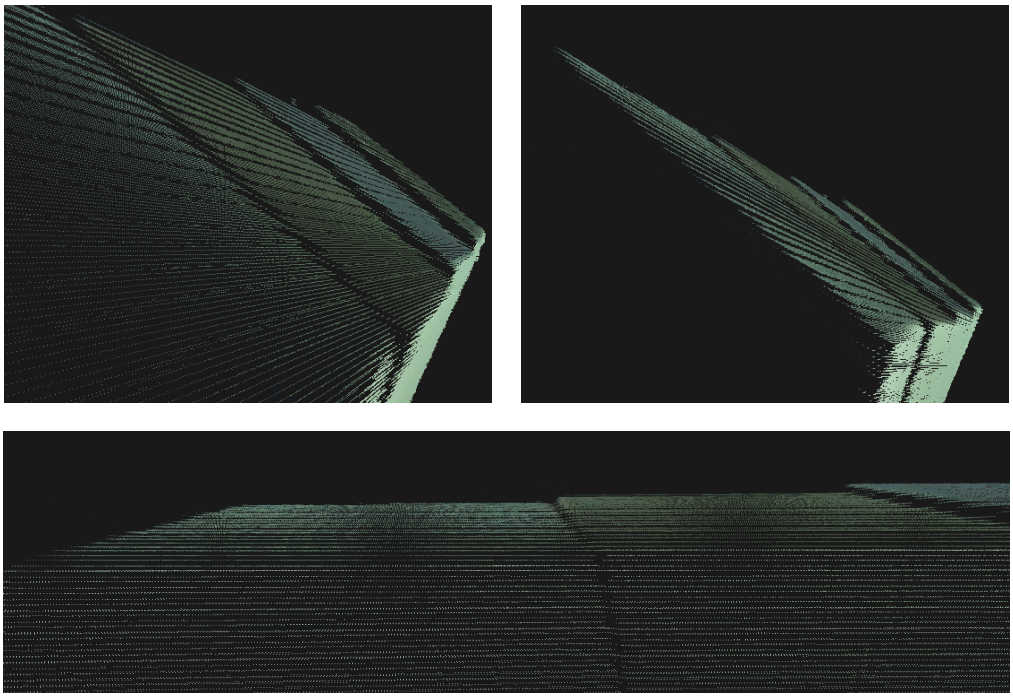


Figure 7.5: *Zoom on the 3D model to prove that we are able to detect even the 1 mm change on the surface of the test cubes.*

For this projector, we see in Figures 7.4 and 7.5 the 3D model we built and a detailed close-up image. As expected, we are able to detect changes of less than 1 millimeter using the Philips cClear XG1 Brilliance as well.

7.2 Efficiency

The time needed for a complete scan should be as minimal as possible. Ideally, we must spend less than 5 minutes building the entire human model; therefore, the time needed to reconstruct a single body part of the patient will be much less.

We have thoroughly analyzed and optimized our application to reduce the time needed to build the 3D model. As a result, it takes around 20 seconds to capture the pictures of the subject with each of the 22 gray-code projected

patterns, and it takes about 40 seconds to reconstruct the 3D model using those 22 pictures.

In the image capture portion, the time needed can be described with the following formula:

$$\textbf{time} = (\textbf{shutter speed} + \textbf{time to change pattern}) \times \textbf{projected patterns}$$

The *shutter speed* depends on how long we want the exposure time to be. For example, the PicoPix 1430 projector has a RGB color wheel in front of the lens that turns very fast to create the illusion of a "white" color; however, it does not turn fast enough. If the shutter speed is on a fast setting, we are able to capture only one color of the color wheel rather than the overlapping effect of the RGB colors. The result is shown in Figure 7.6. To solve this

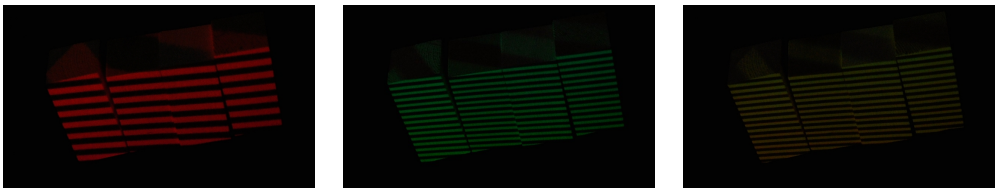


Figure 7.6: *Problem arising if setting a fast shutter speed on the camera when using the PicoPix 1430.*

problem, the only solution is to decrease the shutter speed to 1/25 of a second and increase the ISO sensitivity of the camera. The effect is that the total time needed to capture all 22 pictures increases by 2-3 seconds. The Philips projector does not suffer from this problem, so its shutter speed can be as fast as 1/125 of a second.

Finally, the second important parameter in the previous formula is *time to change pattern*. After every picture is taken, we send the projector a signal to switch patterns and simultaneously put the camera on hold in order to

give the projector enough time to complete its task. If the value of *time to change pattern* is too low, the captured image can be wrong, displaying a combination of the previous pattern with the current one. A good value for *time to change pattern* is 150 milliseconds.

In conclusion, we demonstrated that our scanner takes about 60 seconds to reconstruct a 3D model of a portion of the subject by completing the following tasks: capturing the images, analyzing them to reconstruct the 3D model, creating a wireframe and mesh model from the point cloud and finally, mapping the color information on top of the model. We predict that we will be able to reconstruct the whole human body model in less than 5 minutes by scanning different parts of the patient at the same time and reconstructing them in parallel.

7.3 Costs

Today, the cost of 3D scanners capable of capturing an entire human body ranges from 50K to 300K+ dollars. We want to build an affordable scanner that costs less than $10,000. So far, we are using a computer, a camera and a projector. The cost of the Nikon D5000 is around $400, and the cost of any of the suggested projectors is around $350. With a medium range computer we can build a system with a total cost of around $1200. Using this setup, we capture just one part of the whole subject very well; however, in order to build a complete 3D model of a human being in less than 5 minutes, we need to use more cameras and projectors that can reconstruct different parts of the subject in parallel. After some research (see Section 8.2 for a better insight on the problem), we think we should be able to have a complete 3D model of a person in less than 5 minutes by using a mere 7 cameras, 5 projectors and 1 computer. Therefore, the final cost of the complete setup will be around $5,000 - $6,000.

7.4 Projectors Comparison

So far, we have shown that our human body scanner meets our project requirements, by using either one or the other of the suggested projectors. We discuss in this Section the differences in using those projectors, and we conclude by proving the complete superiority of the Philips cClear XG1 Brilliance over the PicoPix 1430.

Although both projectors meet our requirements, there are two crucial differences described in the two paragraphs that follow.

Color Fidelity By projecting an all-white image on the subject and capturing a picture, we can map the texture on the 3D model. Because our scanning requirements necessitate a very dark room, the light of the projector is the only source of illumination on the subject. If the projector's lamp and color wheel are not very good, the resulting picture will have altered colors that cause the final 3D model to be textured with an inexact color surface. This problem is seen by comparing Figure 7.1 (correct colors), Figure 7.2 and Figure 7.4. The PicoPix 1430 tends to generate colder pictures than the Philips cClear XG1 Brilliance, which has greater color fidelity.

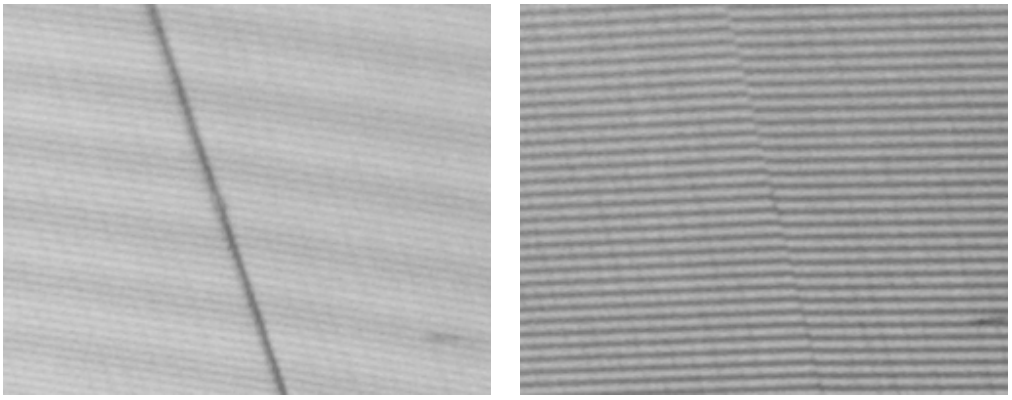
This problem can still be fixed with some post-processing filters on the images; however, it is very important for our scanner to acquire the best color faithful pictures possible, because all the algorithms for skin diseases detection heavily rely on the correctness and accuracy of color information.

Decoding Errors and Artifacts In general, every scanner generates decoding errors when building a 3D model, the origins of which greatly change and range from scattered light problems to subject's movement, picture compression, etc. Because of this, every scanner applies some post-processing algorithm to reduce the amount of errors in its attempt to generate the best 3D model possible; nevertheless, a heavy presence of errors can result in

huge problems, because the application may not be able to remove them all and the final model can be incomplete or worse, completely wrong. We can divide the decoding errors into two subsets: in the first, we include all pixels for which we are not able to assign an intersecting projected plane; in the second, we include all the pixels for which we have assigned an incorrect intersecting plane. We will call the former decoding errors and the latter decoding artifacts.

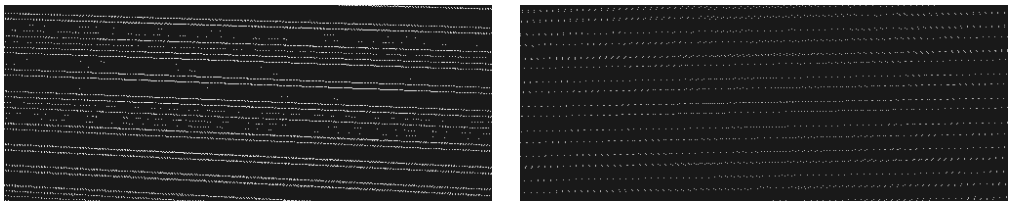
To a certain degree, we would prefer decoding errors to decoding artifacts; the ground truth is that when there are a lot of errors, there are also a lot of artifacts.

As we mentioned, this is the biggest difference we find when using one projector or the other. The reason is that the PicoPix 1430 has too few lumens. In fact, its lamp is not powerful enough even to adequately illuminate the subject, and therefore it is extremely hard to analyze the pictures afterwards. We can better understand this problem in Figure 7.7. Both pictures represent a subject with a projected pattern on top, which has many lines with widths of just 1px. We magnified each picture to the same degree, and so each contains



(a) PicoPix (b) Philips

Figure 7.7: *Zoom on two pictures having projected pattern with lines-width of just 1px.*



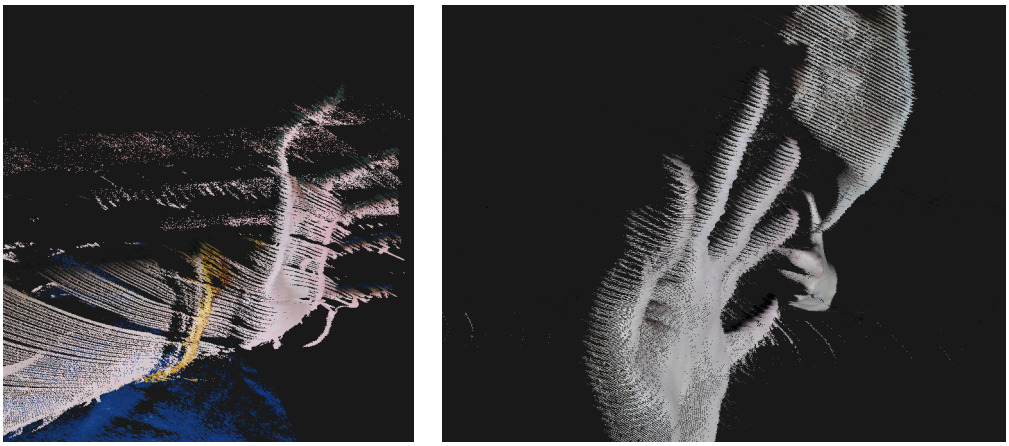
(a) PicoPix (b) Philips

Figure 7.8: *Decoding errors on picture (a) taken with the PicoPix 1430. Here the lines are not evenly spaced as in (b) taken with the Philips projector.*

the same number of horizontal lines; however, in the picture taken using the PicoPix it is very hard to understand which pixels should be considered black and which ones should be considered white. This is a much easier task using the Philips projector. The end result is that some of the lines in the reconstructed model when using the PicoPix are skipped, as evident in Figure 7.8 where we see that the reconstructed model is missing some lines (they are not evenly spaced).

Finally, the PicoPix generates many more decoding artifacts than the Philips projector. The difference is that when the algorithm cannot definitively conclude whether a pixel is white or black, it skips that pixel and does not reconstruct it, thus generating decoding errors. Sometimes, however, the algorithm determines that a pixel is white when it is actually black, or vice versa, and thus generates decoding artifacts. Figure 7.9 effectively displays how many more artifacts are generated when we are using the PicoPix.

We conclude by saying that there are no advantages in using the PicoPix 1430 instead of the Philips cClear XG1 Brilliance other than compactness and marginal monetary savings.



(a) PicoPix (b) Philips

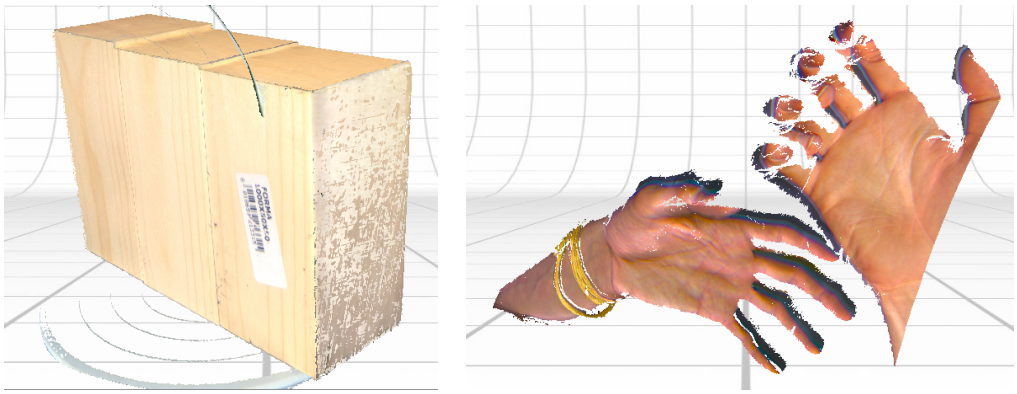
Figure 7.9: *Decoding artifacts in both models; note that the PicoPix one has many more artifacts.*

7.5 Comparison with a Professional Laser Scanner

In this final Section of the Chapter we introduce a professional laser scanner: the NextEngine [Nex]. The NextEngine 3D Scanner captures objects in full color with multi-laser precision. The basic version costs $2,995 and has everything necessary to digitize 3D models, including ScanStudio HD software. It exports to STL, OBJ, VRML, XYZ and other formats. The advanced versions can output 3D scan models to popular design software (like SolidWorks, 3ds Max, and more) and print models on Dimension, 3D Systems, zCorp, Objet, and other 3D printers.

We tested the basic version and have tried to create a model for our test cubes and hand. The results are shown in Figure 7.10. At first glance the results are impressive, but there are some major points worth notice:

- the scan process takes about 4 minutes to scan a very small object, which cannot be further than 43 cm away [GRMB10a];



(a) Test cubes (b) Hands

Figure 7.10: *3D models of the test cubes and hands, both generated with the NextEngine.*

- the laser beam is extremely dangerous when creating the model of a face. Even with the eyelids closed it can still damage the eye's retina;
- it has an accuracy of 0.38 millimeters [GRMB10a], but the subject must remain completely still;
- most of the refinement is done in post-processing; in other words, the 3D point cloud is refined using efficient algorithms to remove decoding errors and artifacts, close the wireframe holes, etc.

After experimenting with the NextEngine Scanner, we have come to the conclusion that it is indeed an impressive tool, the most exceptional aspect being a software that does all the post-processing work, allowing for an easier reconstruction of a 3D model by automatically merging the point clouds and texturing them. If we remove this layer of the NextEngine and merely consider the functions that are implemented in both the NextEngine and our scanner alike (we have not implemented any post-processing refinement algorithms applied to our models, yet), we see that our scanner has much greater potential:

- it can scan much larger surfaces;
- it takes a fraction of the time to scan a subject;
- it is better at compensating for a moving subject;
- it collects more detailed texture information about the model, thanks to our high-resolution camera;
- it can be upgraded with better hardware very easily;
- while not as precise as the NextEngine (<0.38 mm), it is still sufficiently so (<1 mm);
- finally, it costs less than one-third of the price.

We therefore believe that our human body scanner is a strong, well-built tool to use and develop further.

Chapter 8

Future Work

As mentioned in the introduction, the project of building a human body scanner is divided into 4 steps:

- reconstruction of the model of a patient's body part;
- mapping the texture on top of the model;
- merging all models together to generate a complete human body model of the patient;
- registration of the different models taken at different times;
- development of algorithms on top of the models to detect suspicious changes on the patient's skin.

This thesis covers the first two fundamental points, creating a solid base on top of which other developers can build the next steps.

In this Chapter we discuss some promising directions for future research and work. Some of these future paths have already been partially investigated, and we present the results we have come up with so far.

8.1 Dropping Down to 1 Projected Pattern

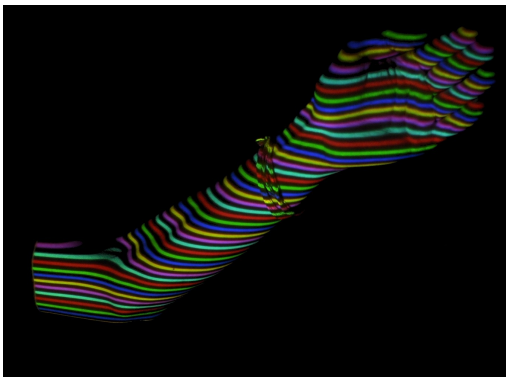
In this thesis, we have discussed our use of a gray code structured light sequence. The benefits of using this approach are many, such as robustness, easier implementation of the decoding algorithm and higher quality of the reconstructed 3D model. Moreover, it only takes us ∼2 minutes to reconstruct the model of a particular subject.

We have been curious, however, about the difference a spatially encoded pattern will make compared to a temporally encoded pattern, like the one we are using. Of course, we are expecting a decrease in the quality of the model, but we would like to better understand if the shorter image-capturing time can compensate for the decreased robustness of this new method [FER07] and [KVG06].

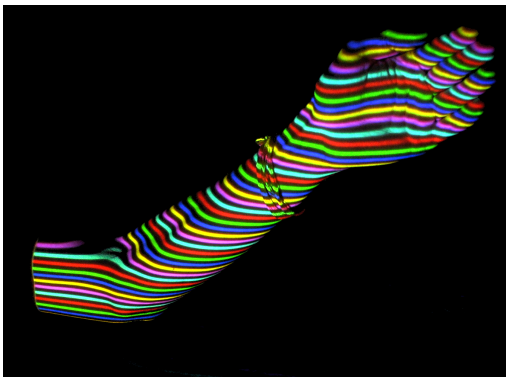
The 1 pattern technique takes full advantage of the potential for any projector to display arbitrary colored images. Here, we need only project a stripe pattern with horizontal lines in fully saturated colors, reducing the search for corresponding pixels to a 1D search along the corresponding scan lines. The projected colors are: red, green, blue, black, cyan, magenta, and yellow. Ideally, to ease the unique assignment of detected stripes to projected ones, we choose a series of stripe colors with a big period by using de Bruijn sequences; however, in our prototype of this 1 pattern technique, we have decided to skip this last step.

Decoding of Colored Stripes The decoding of the colored stripes is done after a preprocessing step. Because a higher contrast leads to better reconstruction results, we apply a vertical sharpening filter to the image. After the preprocessing step, we extract the color of the stripes by applying the procedure shown in Figure 8.1.

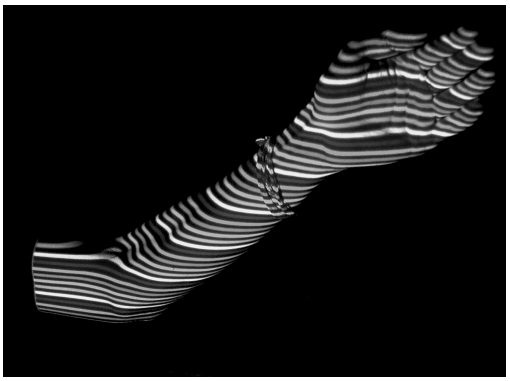
In the last step of Figure 8.1, we decode the colored stripes. Ideally, this should be done using a Dynamic Programming (DP) algorithm that will find



(a) Original image

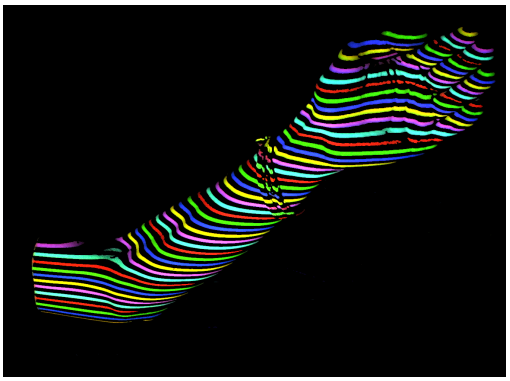


(b) Sharpened image

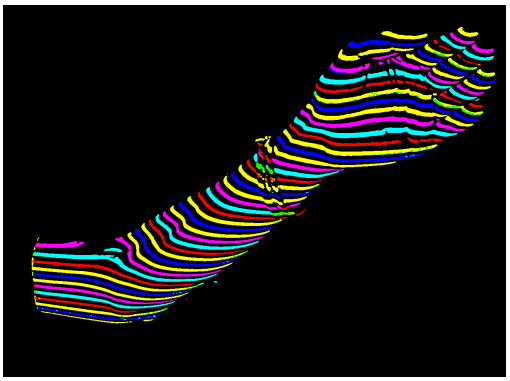


(c) Grayscale image of (b)

(d) B&W image of (c)



(e) Image to decode = (b) mask (d)



(f) Decoded colored stripes

Figure 8.1: *Process to decode a 1 colored pattern.*

the correct match, given that in a de Bruijn sequence of (k, n) a subsequence of length n is present only one time in the whole sequence. If we set $n = 2$ we know the color of a stripe simply by looking at its preceding and following stripe. However, we did not use a de Bruijn sequence, but rather a simple repeating colored sequence. Therefore, in the last step of Figure 8.1 we have only applied a greedy algorithm to choose the color of every pixel based on a simple strategy: to estimate the color of a pixel, the ratios between the channels are examined, i.e. the ratios: $\frac{r}{b}$, $\frac{r}{g}$, $\frac{b}{r}$, $\frac{b}{g}$, $\frac{g}{r}$, $\frac{g}{b}$. In this way, the choice of color assignment is independent of brightness. The sorted ratios will help to decide the resulting color of the pixel by considering larger values first.

We see that our decoded colored stripes image has some errors. In particular, the green stripe is often considered a yellow stripe. This problem can be easily solved using a DP algorithm, as with the one briefly discussed above and in [FER07]. We are very confident that a 1 colored pattern technique is a very promising direction for future research and should be made a priority.

8.2 Full Human Body Scanner

So far, we are only able to reconstruct parts of the patient's body. Our goal it to reconstruct nearly 100% of the skin surface. In order to do so, it is essential to solve the following subproblems.

Patient's Position for Image Capturing We have studied at length different resting positions for the patient in order to determine which produces the best possible scanned results. We concluded that laying the patient on a flat dark surface will help the reconstruction process for two reasons: the dark surface will reduce problems of scattered light and lying down will reduce reconstruction errors, because the patient will remain more still than when

standing up.

Merging Different Surfaces In order to have a complete model of the patient, we need to scan more body parts and then merge them together. The idea is to use multiple cameras and projectors to scan different parts of the front of the patient at the same time, then to turn the patient over onto his or her stomach and scan the back. Finally, we will merge all these parts together. Merging different 3D point clouds can generally be tricky, but knowing the exact location of all the cameras and projectors plus the rough position of the patient allows us to use algorithms, such as the Iterative Closest Point (ICP) algorithm [BM92], to compute a rigid body matching transformation with relatively little effort [LIB].

Simplifying Surfaces The reconstruction of hands and arms, which we use to illustrate our program in this thesis, generates about 800K points, 300K meshes and a $\sim$ 80MB file. It is unfeasible to create a complete 3D model of the patient with such a high definition: the file containing the model would be larger than one gigabyte and the rendering of such a model would become slow and unpleasant. Our only option to solve this problem is to reduce the reconstructed points. This can be done either by deciding not to reconstruct the 3D position of every pixel in the captured images or by applying a post-processing algorithm to simplify the reconstructed surface. Both approaches need to be carefully studied and tested.

8.3 Texture Analysis for Skin Disease Detection

The last step to conquer, which we can approach in parallel with the previous solutions, is the invention and implementation of algorithms to analyze the

patient's skin and detect anomalies in moles or track the evolution of psoriasis. Some research has already been done in our research group; we describe some ideas and techniques in [FPSZ11], [SPM+09] and [FPS11].

Chapter 9

Conclusion

With the rise in awareness of the importance of dermatological examinations, more people have begun to have their skin checked regularly. It is fundamental to provide medical professionals with an adequate healthcare application for tracking at high frequency the evolution of moles and psoriasis on their patients.

In this thesis, we provided a strong foundation for building a powerful human body scanner that is low cost, yet remarkably efficient and accurate in generating a 3D model of any patient. Our human body scanner has the potential to increase the number of dermatological examinations a person has every year, without raising either the costs for health services or the burden on the dermatologists.

In the future, we expect doctors to incorporate our tools and software into their examination routines, to the point where human body scanning becomes a normal procedure for every patient.

Bibliography

[BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.

[FCWC08] Xiuxia Feng, Maoyong Cao, Haixia Wang, and Michael Collier. The comparison of camera calibration methods based on structured-light measurement. *Image and Signal Processing, Congress on*, 2:155–160, 2008.

[FER07] Philipp Fechteler, Peter Eisert, and Jürgen Rurainsky. Fast and High Resolution 3D Face Scanning. In *Proceedings of the 14th International Conference on Image Processing (ICIP2007))*, San Antonio, Texas, USA, 16-19th September 2007. ICIP 2007.

[FPS11] M. Fiorese, E. Peserico, and A. Silletti. Virtualshave: Automated hair removal from digital dermatoscopic images. *IEEE EMBC*, 2011.

[FPSZ11] A. Belloni Fortina, E. Peserico, A. Silletti, and E. Zattra. Where's the naevus? inter-operator variability in the localization of melanocytic lesion border. *Skin Res. and Tech.*, 2011.

[GAVN11] M. Gupta, A. Agrawal, A. Veeraraghavan, and S. G. Narasimhan. Structured light 3d scanning in the presence of global illumina-

tion. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:713–720, 2011.

[gPh] gPhoto - gPhoto Home. `http://www.gphoto.org/`.

[GRMB10a] Gabriele Guidi, Michele Russo, Grazia Magrassi, and Monica Bordegoni. Performance evaluation of triangulation based range sensors. *Sensors*, 10(8):7192–7215, 2010.

[GRMB10b] Gabriele Guidi, Michele Russo, Grazia Magrassi, and Monica Bordegoni. Performance evaluation of triangulation based range sensors. *Sensors*, 10(8):7192–7215, 2010.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[IAT] Image Acquisition Toolbox - MATLAB. `http://www.mathworks.it/products/imaq/`.

[ISM84] S. Inokuchi, K. Sato, and F. Matsuda. Range imaging system for 3-d object recognition. pages 806–808, 1984.

[KVG06] Thomas P. Koninckx and Luc Van Gool. Real-time range acquisition by adaptive structured light. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(3):432–445, March 2006.

[Lau94] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2):150–162, February 1994.

[Lau95] Aldo Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(2):188–195, February 1995.

[LIB] Libicp: Library for iterative closest point fitting. `http://www.cvlibs.net/software/libicp.html`.

[LT09] Douglas Lanman and Gabriel Taubin. Build your own 3d scanner: 3d photograhy for beginners. In *SIGGRAPH '09: ACM SIGGRAPH 2009 courses*, pages 1–87, New York, NY, USA, 2009. ACM.

[Mat] MATLAB - The Language of Technical Computing. `http://www.mathworks.it/products/matlab/`.

[Mel] Melanoma education foundation. `http://www.skincheck.org/`.

[Nex] Nextengine 3d laser scanner. `http://www.nextengine.com/`.

[NSM+94] F Nachbar, W Stolz, T Merkle, A B Cognetta, T Vogt, M Landthaler, P Bilek, O Braun-Falco, and G Plewig. The abcd rule of dermatoscopy. high prospective value in the diagnosis of doubtful melanocytic skin lesions. *Journal of the American Academy of Dermatology*, 30(4):551–559, 1994.

[Opea] OpenCV Wiki. `http://opencv.willowgarage.com/`.

[Opeb] OpenGL - The Industry Standard for High Performance Graphics. `http://www.opengl.org/`.

[PA82] J L Posdamer and M D Altschuler. Surface measurement by space-encoded projected beam systems. *Computer Graphics and Image Processing*, 18(1):1–17, 1982.

[RCM+01] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno. A low cost 3d scanner based on structured light, 2001.

[SOY+09] Ryusuke Sagawa, Yuya Ohta, Yasushi Yagi, Ryo Furukawa, Naoki Asada, and Hiroshi Kawasaki. Dense 3d reconstruction method using a single pattern for fast moving object. In *Proc. 2009 IEEE 12th International Conference on Computer Vision*, pages 1779–1786, Kyoto, Sep. 2009.

[SPB04] Joaquim Salvi, Jordi Pagès, and Joan Batlle. Pattern codification strategies in structured light systems. *PATTERN RECOGNITION*, 37:827–849, 2004.

[SPM+09] A. Silletti, E. Peserico, A. Mantovan, E. Zattra, A. Peserico, and A. Belloni Fortina. Variability in human and automatic segmentation of melanocytic lesions. *IEEE EMBC*, 2009.

[TW07] Philip Treleaven and Jonathan Wells. 3d body scanning and healthcare applications. *Computer*, 40:28–34, 2007.

[VRM] Vrml97 and related specifications. `http://www.web3d.org/x3d/specifications/vrml/`.

[Zha99] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *Computer Vision, IEEE International Conference on*, 1:666, 1999.

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

List of Figures

List of Tables