Alma Mater Studiorum · Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA
Department of Electrical, Electronic and Information Engineering
"Guglielmo Marconi"
Automation Engineering M

DEVELOPMENT OF A PHOTOMETRIC STEREO SYSTEM

Candidate:
Andrea Campisi

Supervisor:
Chiar.mo Prof.
Luigi Di Stefano

Co-advisor:
Ing. Luca Piccinini

Academic year 2016/17
Session III

"Quisque faber suæ ipsæ fortunæ"
Appius Claudius Cæcus

Introduction

Photometric stereo is a method of estimating surface geometry by using a fixed-position camera and multiple light sources.

It was proposed by Robert J Woodham in 1980 as an alternative to other stereo techniques, that needed two images of the same object, viewed from different directions, to determine its surface orientation. These techniques require some knowledge of the correspondence between picture elements, that is not always easy to achieve.

The idea of photometric stereo is to vary the direction of incident illumination between successive images, while the viewing direction is constant; in this way, the correspondence of image points is known beforehand, as the position of the object is not changed, and surface orientation is determined by using the radiance values recorded with the different images.

Photometric stereo has several applications: it is used to detect small surface defects, imperceptible to the human eye; it has also medical applications, i.e. to detect skin lesions and help in the diagnosis of melanoma. More recently, it has been used also to perform a non-invasive 3D scan of surfaces, and as a cheap method to get a real-time facial motion capture, widely used in computer graphics applications.

The goals of this thesis are to describe the main algorithms used to achieve a photometric stereo analysis of surfaces, and the realization of a hardware device capable of acquire the images.

In Chapter 1 we will introduce the methods currently used to perform 3D scanning.

In Chapter 2 we will discuss photometric stereo in detail, explaining the most popular algorithms and the methods used to get light calibration, height map and curvature maps.
In Chapter 3 we will show two different implementations of Woodham's algorithm, one of them performed using a GPU.
In Chapter 4 we will describe the hardware choices that led us to build the rig used to acquire the images.
In Chapter 5 there will be the results of our work, and a comparison with a commercial software (MVTec Halcon).

Contents

Introduction **i**

1 State-of-the-art of 3D scanning techniques **1**

2 Photometric stereo **4**

2.1 Woodham's algorithm 4

2.2 Hayakawa's uncalibrated algorithm 9

2.3 Light calibration 11

2.4 Getting the height map 13

2.5 Curvature analysis 16

2.6 Limitations of photometric stereo 18

3 Algorithm implementation **21**

3.1 CPU implementation 21

3.1.1 Light calibration 23

3.1.2 Photometry 24

3.2 GPU implementation 25

4 Image acquisition system **27**

4.1 Camera, light source and controller 27

4.2 Image acquisition 29

4.3 Software User Interface 29

5 Results **31**

5.1 Light calibration 31

5.2 Photometry results 32

6 Conclusions 38

Chapter 1

State-of-the-art of 3D scanning techniques

One of the most classic and challenging problems in computer vision is shape recovery that, altogether with surface analysis, has several applications in industrial productions, medicine and movie industries. During years several methods have been proposed, that let us recover the shape and express it as a depth map, as a surface normal map or as a surface gradient map. These methods can be classified according to the interaction between the scanner and the object, and a tree-diagram of this classification is shown in Figure 1.1.

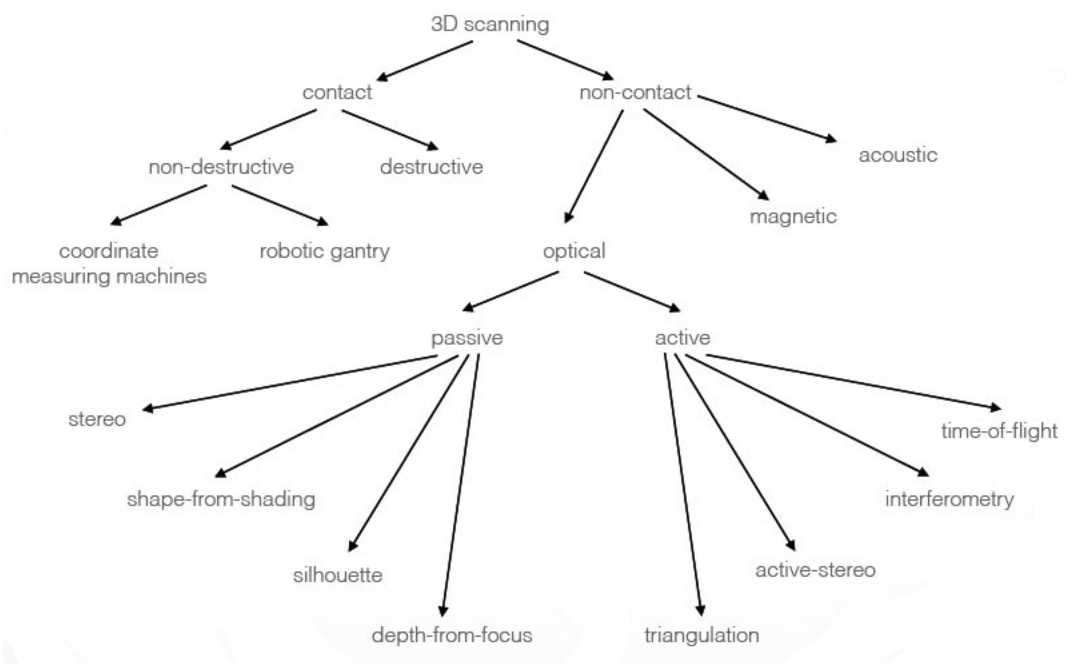


Figure 1.1: Tree diagram showing all the 3D scanning techniques

The first optical and passive approach was the so-called shape from shading (SFS), introduced by Horn in 1975: he thought it was possible to retrieve information about surface by using one single image. Now, it is known to be an ill-posed problem, as it can provide non-unique solutions and is affected by convex/concave ambiguity, as shown in Figure 1.2.

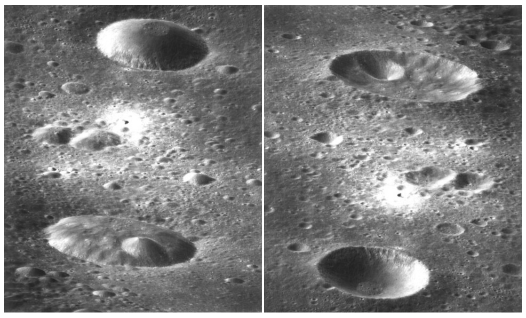


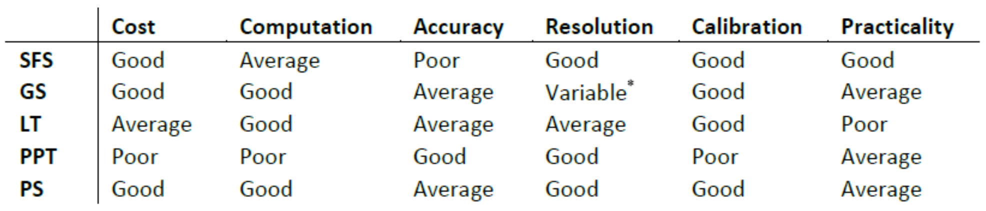
Figure 1.2: Image showing the crater illusion. In the left image there appear two domes, while in the right one (that is the same upside down) there are two craters.

Later, in 1980, Woodham proposed the photometric stereo (PS) approach, based on the fact that the amount of light reflected by an object is linked both to the orientation of its surface and to the direction of incoming light. In 1994, Hayakawa showed how to estimate a surface normal and reflectance map without a-priori knowledge about light direction or intensity (mandatory for Woodham's algorithm). Other popular approaches are geometric stereo, laser triangulation and projected pattern triangulation.
Geometric stereo (GS) is based on the usage of two cameras, placed at a known distance one from the other; the acquired images are rectified to ease the search of corresponding points, then the coordinates of the matches have to be find in order to get a point cloud. Geometric stereo can provide good results, but it is strongly dependent on the correspondence density.

Laser triangulation (LT) is based on the usage of a system composed of a laser and a camera; while the laser projects a line onto an object, the camera acquires the images. Then information about the height of the object are extracted by deploying the deformation of the projected line.

Projected pattern triangulation (PPT) is similar to laser triangulation, as it is based on the projection of a known pattern onto an object; the 3D shape of the object is retrieved after computing the deformation of the pattern. This is a very accurate procedure, but also very expensive.

Among all these techniques, photometric stereo can be considered as the best compromise between cost and accuracy. In Figure 1.3 there is a table showing the main differences in terms of cost, computation time, accuracy and other qualities between the methods aforementioned.



	Cost	**Computation**	**Accuracy**	**Resolution**	**Calibration**	**Practicality**
SFS	Good	Average	Poor	Good	Good	Good
GS	Good	Good	Average	Variable*	Good	Average
LT	Average	Good	Average	Average	Good	Poor
PPT	Poor	Poor	Good	Good	Poor	Average
PS	Good	Good	Average	Good	Good	Average

Figure 1.3: Table showing a comparison between optical 3D scanning alternatives

Chapter 2

Photometric stereo

In this chapter, we will describe the main algorithms used in photometric stereo, that are Woodham's and Hayakawa's algorithms. We will also describe the procedure to calibrate the light sources, required for our implementation. In section 4 we will describe a way to integrate the normal map to get the height map, used for 3D reconstruction. In section 5 there is a description of how to get surface curvature information, from the normal map. In section 6 we will list the main limitations of photometric stereo, and some ways to overcome them.

2.1 Woodham's algorithm

First, let us give some basic definitions.
The irradiance is a measure of light arriving at a surface. It can be expressed as $L(\lambda, x_p, N_p)$, as a function of the wavelength of the light, the point where the light is directed to, and the normal direction of the surface on that point. Unit of measurement: $[W/m^2]$
The radiance is a measure of light emitted or reflected from a surface, and depends on the direction of the surface. It can be expressed as $E(\lambda, x, d)$ as a function of the wavelength of the light, the point from which is emitted the radiance, and the emitting direction. Unit of measurement: $[W/(sr * m^2)]$

The reflectance is the ratio between the amount of light reflected and the amount of light incident to an object. It is an adimensional parameter, and it depends on the material of which an object is made. It can be expressed as a function $\phi(i, e, g)$ of the incident (i), emergent (e) and phase (g) angles. The luminance is the ratio between the intensity of the light emitted by a source to an observer and the apparent area of the emitting surface, as seen by the observer. Unit of measurement: $[cd/m^2]$

A BRDF (Bidirectional Reflectance Distribution Function) is a 4-dimensional equation that describes how a light is reflected at an opaque surface. In general, it is given by the ratio between the emitted radiance and incoming irradiance on a point. If vectors ω_0 and ω_i represent the direction of the light reflected to an observer and the direction of incoming light respectively, then the BRDF can be defined as:

$$f_r(\omega_0, \omega_i) = \frac{dL_r(\omega_0)}{dE_i(\omega_i)} = \frac{dL_r(\omega_0)}{L_i(\omega_i) cos\theta_i d\omega_i} \tag{2.1}$$

The Lambertian reflectance is a property that defines a diffusely reflecting surface: the apparent luminosity of a Lambertian surface is the same regardless of the observer's angle of view. The luminance of the surface is isotropic and the luminous intensity obeys Lambert's cosine law.

The behaviour of a Lambertian material and an example of a Lambertian sphere are shown in Figure 2.1.

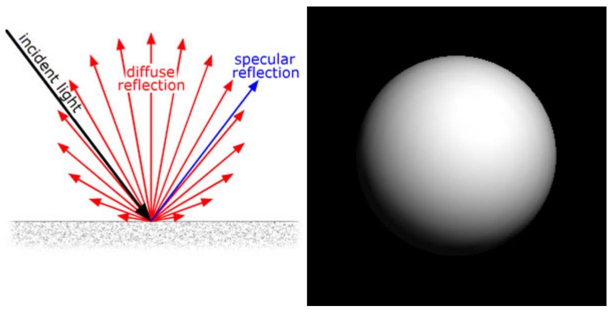


Figure 2.1: In the left image, diffuse and specular reflection from a lambertian surface. In the right one, an example of a lambertian sphere

Now, consider an object whose surface can be written as $z = f(x, y)$; a surface normal can be defined as $n = [\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, -1]$.
Defining $p = \frac{\partial z}{\partial x}$ and $q = \frac{\partial z}{\partial y}$ then

$$n = [p, q, -1] \tag{2.2}$$

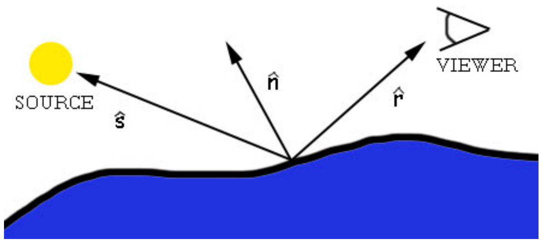


Figure 2.2: Basic diagram of light reflection

Consider figure 2.2. Define vectors $\hat{n}$ and $\hat{s}$ as the surface normal vector and the vector that points to the light source respectively. They can be defined as follows:

$$\hat{n} = [p, q, -1]^T \tag{2.3}$$

$$\hat{s} = [p_s, q_s, -1] \tag{2.4}$$

Our problem is to determine the direction of the surface normal, knowing the direction of the lights. Let us assume that:

1. The surface has a Lambertian reflectance
2. Light rays are parallel and uniform
3. The camera is orthographic and monochromatic

Then the Lambertian BRDF is a constant:

$$BRDF_{Lamb} = \frac{\rho_d}{\pi} \tag{2.5}$$

and the surface radiance is given by:

$$E = \frac{\rho_d}{\pi}\hat{n} \cdot \hat{s} = \rho\hat{n} \cdot \hat{s} = \rho\hat{s}^T \cdot \hat{n} \tag{2.6}$$

Suppose that the irradiated light is directed towards a camera, and that the response of the camera is linear; In this case, the radiance coincides with the intensity of a pixel:

$$I = \rho \hat{n} \cdot \hat{s} \tag{2.7}$$

The problem is that this is an equation with 3 known variables (the 2 components of light's direction and the pixel intensity, namely p_s, q_s, I) and 3 unknowns (the components of the normal and the albedo, p, q, ρ).

Consider a simple case, using Lambert's cosine law:

$$E = \rho L \cos(\theta) \tag{2.8}$$

where E is the radiance, ρ is the albedo of the surface, L is the irradiance and θ is the angle between the normal and the direction of light. The value of $\cos(\theta)$ can be retrieved by applying the scalar product definitions from linear algebra:

$$\hat{n} \cdot \hat{s} = pp_s + qq_s + 1 = \sqrt{p^2 + q^2 + 1}\sqrt{p_s^2 + q_s^2 + 1}\cos(\theta) \tag{2.9}$$

This leads to:

$$E = \rho L \frac{pp_s + qq_s + 1}{\sqrt{p^2 + q^2 + 1}\sqrt{p_s^2 + q_s^2 + 1}} \tag{2.10}$$

If the response of the camera is linear, then the intensity of the pixel coincides with the irradiance, then:

$$I = \rho L \frac{pp_s + qq_s + 1}{\sqrt{p^2 + q^2 + 1}\sqrt{p_s^2 + q_s^2 + 1}} \tag{2.11}$$

This is an equation with 3 knowns (p_s, q_s, I) and three unknowns (p, q, ρ). So we need at least three images to solve this problem.
Define $\overline{I} = [I_1, I_2, I_3]^T$ as the column vector collecting the intensities recorded at a given point (x, y) in the three images I_1, I_2 and I_3.
Define $N = [\hat{n_1}, \hat{n_2}, \hat{n_3}]^T$ as the vector containing the normals in point (x, y) computed in the three images.

Define $L = [s_1, s_2, s_3]$ as the vector collecting the directions of incoming light at a point (x, y) in the three images.

Equation 2.7 becomes:

$$\overline{I} = \rho N \cdot L = \rho L^T \cdot N \tag{2.12}$$

Then

$$N = \rho^{-1}(L^T)^{-1}\overline{I} \tag{2.13}$$

$$\rho = ||(L^T)^{-1}\overline{I}|| \tag{2.14}$$

This computation requires that matrix L must be non-singular (in other words the directions of the lights must not lie on the same plane). This means that in the three images the orientation of light must be changed. Now, consider a more general case in which we get n images. This means that matrix L^T is not a square matrix, but we can refer to the Moore-Penrose pseudoinverse. Equations 2.13 and 2.14 become:

$$N = \rho^{-1}(L^T L)^{-1}L^T\overline{I} \tag{2.15}$$

$$\rho = ||(L^T L)^{-1}L^T\overline{I}|| \tag{2.16}$$

Note that equations 2.15 and 2.16 represent the solution of the photometric stereo problem by means of the Least Squares method.

Remark. *Woodham's algorithm assumes that the camera performs an orthographic projection. That is, you must use a telecentric lens or a lens with a long focal distance. It also assumes that each of the light sources delivers a parallel and uniform beam of light. That is, you must use telecentric illumination sources with uniform intensity or, as an alternative, distant point light sources. Additionally, the object must have lambertian reflectance characteristics, i.e., it must reflect incoming light in a diffuse way. Objects or regions of an object that have specular reflectance characteristics (i.e., mirroring or glossy surfaces) cannot be processed correctly and thus lead to erroneous results.*

2.2 Hayakawa's uncalibrated algorithm

In 1994 Hayakawa showed how to estimate a surface normal and reflectance without a-priori knowledge on the light direction or intensity. This method is based on the Singular Value Decomposition (SVD) of the matrix collecting pixel intensities; it assumes a Lambertian reflectance model (the same as the Woodham's algorithm), but adds also some constraints on the surface reflectance (at least 6 pixels must have a known or constant reflectance) and on the light-source intensity (at least 6 frames must have a known or constant light intensity).
Consider a $p \times f$ matrix I:

$$I = \begin{bmatrix} i_{11} & \cdots & i_{1f} \\ \vdots & \vdots & \vdots \\ i_{p1} & \cdots & i_{pf} \end{bmatrix} \tag{2.17}$$

that is the matrix collecting the pixel intensities for each frame acquired. Under the assumption of Lambertian reflectance for the object, and supposing the light source is single and far from it, then:

$$I = RNMT \tag{2.18}$$

where R is the reflectance matrix, N is the normal matrix, M is the light-direction matrix and T is the light-intensity matrix. Equation 2.18 can be written in a compact form:

$$I = SL \tag{2.19}$$

where S is the surface matrix, L is the light-source matrix.

Assuming that $p \geq f$, matrix I can be decomposed into three matrices $U_{p\times f}$, $\Sigma_{f\times f}$ and $V_{f\times f}$:

$$I = U\Sigma V \tag{2.20}$$

Supposing that the image I we get from the camera could be affected by noise (thus it would be different from the noiseless ideal one I^*), we can rely on an estimation of the ideal image starting from the best rank-3 approximation; let $U^{'}$,$\Sigma^{'}$,$V^{'}$ be the first three columns of U, the first 3×3 submatrix of Σ and the first 3 rows of V respectively; then the estimation of I^* is:

$$\hat{I} = U^{'}\Sigma^{'}V^{'} \tag{2.21}$$

From this equation we can retrieve the estimations of the surface and light-source matrices:

$$\begin{aligned} \hat{S} &= U^{'} \pm [\Sigma^{'}]^{\frac{1}{2}} \\ \hat{L} &= \pm[\Sigma^{'}]^{\frac{1}{2}}V^{'} \end{aligned} \tag{2.22}$$

The different signs in the previous equations correspond to the solutions in the right- and left-handed coordinate system. We can choose the solution in the right-handed coordinate system from the two solutions in equation 2.22 by using the relationship of an arbitrary triplet of light-source directions, which do not lie in a plane. In general, matrices $\hat{S}$ and $\hat{L}$ are different than S and L; we can define a matrix A such that

$$\begin{aligned} S &= \hat{S}A \\ L &= A^{-1}\hat{L} \end{aligned} \tag{2.23}$$

A can be found by applying the constraints on the surface reflectance or on the light intensity described at the beginning of this section.

2.3 Light calibration

Light calibration is a sub-problem linked to the photometric stereo. In fact, the orientation of lights with respect to the camera must be known in order to solve the equations to determine normals and albedo. A widely-used technique to perform light calibration is using a reflective sphere, whose radius and position (with respect to the camera) must be known; the light source direction is determined thanks to the highlight present on the surface of the sphere. In Figure 2.3 it is shown the reflection on a black sphere when light source has different positions.

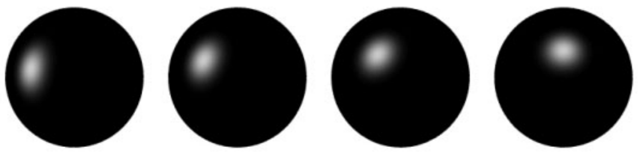


Figure 2.3: Image showing different highlight locations due to different light positions

The idea is to place a highly reflective sphere in the scene; it should act as a perfect mirror, reflecting the incoming light towards the viewer. For a perfect mirror, light direction can be computed as follows:

$$L = 2(N \times R) \times N - R \tag{2.24}$$

where L is the light direction, N is the normal direction and R is the reflection direction. Those vectors are shown in Figure 2.4.
To solve equation (2.24), we need R and N.

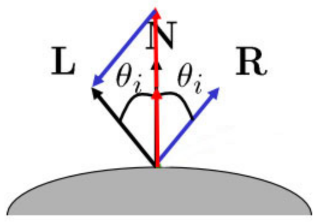


Figure 2.4: Diagram of specular reflection

As we are considering the reflection coming from a highlight directed to the camera, then R is the vector pointing to the camera, that is $[0, 0, 1]$. To find N, we need to determine the direction of the normal vector at the point with maximum intensity on the sphere. First, we have to find the region with the maximum intensity; it can be seen as a tilted circle. Then we can find its central point. Consider Figure 2.5. Knowing the positions of the center of the circle $[C_x, C_y]$ and of the center of the sphere $[S_x, S_y]$ on the image, we can write:

$$N_x = C_x - S_x \tag{2.25}$$

$$N_y = C_y - S_y \tag{2.26}$$

By applying the Pythagorean theorem:

$$N_z = \sqrt{r^2 - N_x^2 - N_y^2} \tag{2.27}$$

where r is the radius of the sphere. N_x, N_y, N_z are the components of the normal vector.

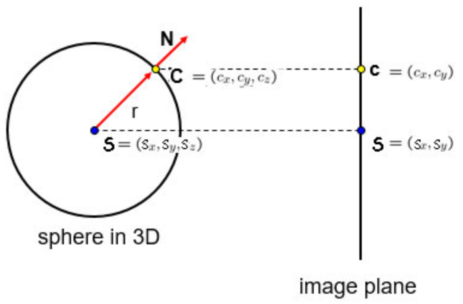


Figure 2.5: Projections of center of the sphere and of the circle into the image plane

2.4 Getting the height map

Once we get the normal map and the albedo, we can get some information about the depth of the object. Theoretically, if we integrate point-by-point the normal map, we would get the depth map.

The simplest approach is the following. Recall the definition of vector n:

$$n = [\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, -1] = [p, q, -1] \tag{2.28}$$

where z is the surface of our object. Once we get the estimations of p and q, then:

$$z(x, y) = z(x_0, y_0) + \int_{(0,0)}^{(\bar{x},\bar{y})} (pdx + qdy) \tag{2.29}$$

In a practical way, we can integrate p along a row of the image to get $z(x, 0)$, then integrate q along all the columns starting from the value of the first row (as shown in Figure 2.6).

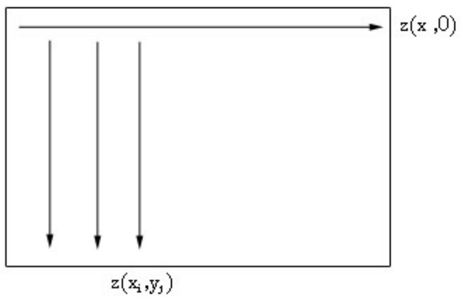


Figure 2.6: Sketch of the integration procedure

If one integrates a derivative field along a closed curve, the final value should coincide with the starting one, but the problem is that p and q can be noisy, meaning that the final result would be affected by an additive error. This is clearly a problem involving the integrability condition of the estimate of the normal field.
In general, given a function $f = f(x, y)$ the integrability holds if:

$$\frac{\partial}{\partial y}\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}\frac{\partial f}{\partial y} \tag{2.30}$$

In our case

$$\frac{\partial p}{\partial y} \neq \frac{\partial q}{\partial x} \tag{2.31}$$

B.K.P. Horn[5] in 1986 proposed a solution based on the minimization of a functional cost:

$$\iint_{image} (z_x - p)^2 + (z_y - q)^2 dxdy \tag{2.32}$$

where z_x and z_y are the partial derivatives of the surface and p and q are the estimated components of the gradient.
A variant of the solution was proposed by Frankot and Chellappa [6], considering the surface z as:

$$z(x, y) = \sum_{\omega \in \Omega} C(\omega)\phi(x, y, \omega) \tag{2.33}$$

In this way, the problem is finding the expansion coefficients $C(\omega)$ that minimize equation 2.32. Both the solutions provide values of z_x and z_y such that the resulting surface z is integrable.
However, a simpler method to get the height map is the following [15].
Recall the definition of surface normal:

$$n = [\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, -1] = [n_x, n_y, n_z] \tag{2.34}$$

and of p and q:

$$p = \frac{n_x}{n_z} \tag{2.35}$$

$$q = \frac{n_y}{n_z} \tag{2.36}$$

As we are considering discrete surfaces, we can approximate them using finite difference expressions:

$$p \approx z(x + 1, y) - z(x, y) \tag{2.37}$$

$$q \approx z(x, y + 1) - z(x, y) \tag{2.38}$$

Then, combining the previous relations, we get:

$$n_x = n_z[z(x+1,y) - z(x,y)] \tag{2.39}$$

$$n_y = n_z[z(x,y+1) - z(x,y)] \tag{2.40}$$

To find the values of the depth, we have to solve those linear equations, in a least square sense. Note that we have a couple of equations for each pixel of the image; we have to build up a linear system, and the final equation will have a form like $Az = v$, where:

1. matrix A will be a sparse matrix, in which each row will represent a constraint for each point inside the object and on its edge. If the normal map has $npixel$ pixels, then A will have a size of $(2 \cdot npixel, npixel)$;

2. vector v will contain all the values of partial derivatives p and q.

$$Av = z \rightarrow \begin{bmatrix} 1 & -1 & 0 & 0 & \cdots \\ 1 & 0 & -1 & 0 & \cdots \\ 1 & 0 & 0 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} z(x,y) \\ z(x+1,y) \\ z(x,y+1) \\ \vdots \end{bmatrix} = \begin{bmatrix} -p \\ -q \\ \vdots \end{bmatrix} \tag{2.41}$$

Solution can be found by applying the least square estimation method:

$$z = (A^T A)^{-1} A^T v \tag{2.42}$$

2.5 Curvature analysis

In general, curvature is a property on an object; it intuitively represents the deviation of a surface from being flat or, in a two-dimensional case, of a curve from being straight. In mathematics, there are several definitions of curvature, depending on the context. As for surface analysis, it is clear that the study of curvature is important because it can show if and where eventual anomalies are present. In image processing, curvature analysis is used to segment scratches or bumps on a surface.

On a bidimensional surface, curvature varies depending on the direction along which it is computed. Given a point P on the surface, we have to consider all planes passing by the normal vector entering the surface in P; the intersection between each plane and the surface determines a plane curve on which it is possible to compute the curvature. It is positive if the curve follows the same direction of the normal, negative otherwise.
The maximum and minimum values of the curvature are said *main curvatures*; given the two main curvatures k_1 and k_2, it is possible to determine two measures of the curvature: *Gaussian and mean curvature.*
Let us define a bidimensional vector field $f(r, c)$.
Gaussian curvature K is the product of the two principal curvatures, and can be computed as follows:

$$K = \frac{\frac{\partial f(r,c)^2}{\partial^2 r} \cdot \frac{\partial f(r,c)^2}{\partial^2 c} - \frac{\partial f(r,c)^2}{\partial r \partial c} \cdot \frac{\partial f(r,c)^2}{\partial c \partial r}}{(1 + \frac{\partial f(r,c)^2}{\partial r} + \frac{\partial f(r,c)^2}{\partial c})} \tag{2.43}$$

If $K > 0$ then the surface is said to have an elliptic point, where the surface is dome-like; if $K < 0$ the surface has a saddle point, while if $K = 0$ it is a parabolic point.
Mean curvauture H is the arithmetic average of the two main curvatures. It can be computed as follows:

$$H = \frac{A - B + C}{D} \tag{2.44}$$

where:

$$A = (1 + \frac{\partial f(r,c)^2}{\partial r})\frac{\partial^2 f(r,c)}{\partial c^2} \tag{2.45}$$

$$B = \frac{\partial f(r,c)}{\partial r}\frac{\partial f(r,c)}{\partial c}(\frac{\partial^2 f(r,c)}{\partial r \partial c} + \frac{\partial^2 f(r,c)}{\partial c \partial r}) \tag{2.46}$$

$$C = (1 + \frac{\partial f(r,c)^2}{\partial c})\frac{\partial^2 f(r,c)}{\partial r^2} \tag{2.47}$$

$$D = (1 + \frac{\partial f(r,c)^2}{\partial r} + \frac{\partial f(r,c)^2}{\partial c})^{\frac{3}{2}} \tag{2.48}$$

2.6 Limitations of photometric stereo

As already specified in Section 2.1, Woodham's photometric stereo algorithm relies on some hypothesis, that here we recall:

1. The surface has a Lambertian reflectance
2. Light rays are parallel and uniform
3. The camera is orthographic and monochromatic

When one of those conditions is not met, results can not be considered as reliable. The strongest limitation is represented by the reflectance of surfaces: in fact, often it is required to analyze metallic or plastic objects, that are not Lambertian. In these cases, the presence of highly reflective points may affect negatively the photometric process. It is possible to use other reflectance models, i.e. Phong model, that takes into account also ambient and specular lighting as shown in Figure 2.7.

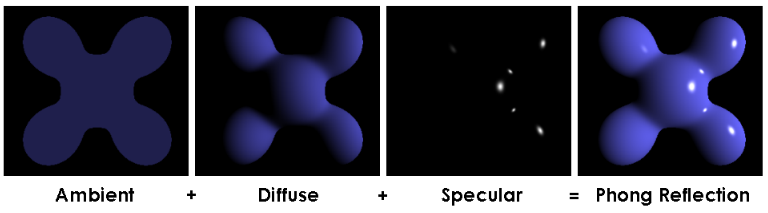


Figure 2.7: Visual illustration of Phong equation

To solve the problem of the presence of complex global illumination effects, a solution has been proposed by several authors in [9] that lets us "decompose" an image and get the direct and global illuminations. Direct illumination is due to the light source, while the global is due to the reflection, refraction and diffuse lights coming from other points.
Their method, depicted in Figure 2.8, consists in projecting a high-frequency moving illumination pattern on the scene (i.e. a shifted checker board pattern) that will create "patches" on the scene: some of them will be lit, the

others will be unlit. The lit patches will contain both global and direct illumination while the unlit ones will contain only the global illumination. The authors also describes a way to get direct and global illumination by using a single image.
This method should be applied to each acquired photometric stereo image.

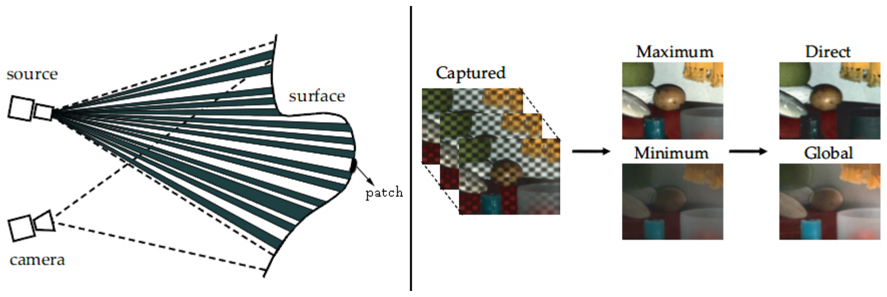


Figure 2.8: In the left image it is shown the basic setup of camera, projector and a surface; in the other there is an example of the acquired images and the final results.

Another limitation is the camera resolution. In general, it affects the size of the final retrieved 3D scan of an object. Furthermore, a surface can have sub-pixel geometric structures that unlikely will be well-described by a single normal per pixel. The best idea would be increasing the camera resolution, to improve the level of detail of the acquired image and of the final model, but this has the drawback of increasing the computation load.
In 2016 a system employing a ultra-high resolution camera was tested [10]. It is composed of a Microsoft Apsara camera (resolution of 1.6 gigapixel) and a set of 16 controllable LEDs. Among the results, it was shown how this device could scan a painting (from a distance of 1.6 meters) and, through the photometric stereo technique, determine a normal map at brush-stroke detail (see Figure 2.9).

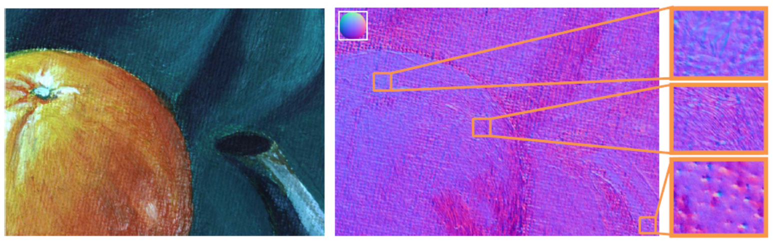


Figure 2.9: Result of the Gigapixel apparatus

In [11] it is shown a procedure to get a high-resolution model of an object starting from a lower resolution set of images. At first, to recover the distribution of surface normals inside each pixel it uses a generalized reflectance model. The normal distribution is used to infer sub-pixel structures on a surface of uniform material. Figure 2.10 shows a result of this approach.

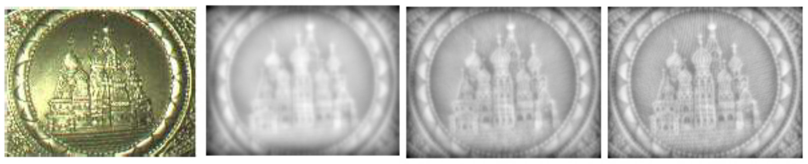


Figure 2.10: Result of the resolution-enhanced algorithm. The first image is the acquired frame, the second is the depth field retrieved from the first; the successive ones are the depth maps retrieved with the enhancing at different levels (2x and 4x)

Chapter 3

Algorithm implementation

In this chapter we will discuss how Woodham's algorithm has been implemented through a popular open source computer vision library (OpenCV), explaining also how we implemented the calibration algorithm. Moreover, we will also show the implementation of the algorithm using a parallel computing platform (CUDA) to make the process faster.

3.1 CPU implementation

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision [19]. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

One of the basic classes of this library is Mat, the matrix class. It represents an n-dimensional dense numerical single-channel or multi-channel array. It

can be used to store real or complex-valued vectors and matrices, grayscale or color images, vector fields, point clouds and many other structures. The Mat class supports also algebraic operations (i.e. inverse and transpose of a matrix). In OpenCV one can find several useful and already-implemented computer vision algorithms, for example ones regarding image thresholding, segmentation and morphologic analysis.
Our implementation of photometric stereo is divided in three principal phases:

1. Light calibration;
2. Image acquisition;
3. Photometry.

In the following, we will describe light calibration and the basic implementation of Woodham's algorithm. As for image acquisition, it will be described in Chapter 4, altogether with the image acquisition system.
The results of photometry can later be used for other processes, i.e. curvature analysis and depth computation. We have implemented both Gaussian and mean curvature derivation, and results will be shown in Section 5. As for the depth map, we used a Matlab implementation of the procedure explained at the end of Section 2.4 [20].

3.1.1 Light calibration

The basic concept of light calibration is explained in Section 2.3. Briefly, the idea is to place a black reflective ball in front of the camera, and capture a frame each time a light is on. Then, we have to find the most reflective point on the sphere for each acquired image. As the position of the sphere will remain constant during calibration process, we can make use of an alpha mask; applying it to each acquired image, we get almost-black images, in which the only white regions are the highly reflective ones. In this way, by applying the well-known morphological operation of opening by an elliptical structuring element, it is possible to locate the reflective spots, and their coordinates. Finally, by applying equations from (2.25) to (2.27), we can find the vector that defines the orientation of a light source.

Algorithm 1 Light calibration pseudocode

```
1:  Load Mask image;
2:  Apply opening by a circular structuring element;
3:  Individuate the bounding box, then get its side length (that coincides
    with the radius of the sphere) and its center $(S_x, S_y)$;
4:  Load all acquired images;
5:  for all images do
6:      Apply the mask image. The result will be a black image, with a white
        area representing the reflection area;
7:      Apply opening by a circular structuring element;
8:      Determine the bounding box and its center $(C_x, C_y)$;
9:      Apply equations (2.25) to (2.27) and get the components of normal
        vector $N$;
10:     Save $N$ inside a matrix (Light matrix);
11: end for
12: Export the light matrix.
```

3.1.2 Photometry

The main problem to solve is the construction of equations (2.15) and (2.16). In fact, to make our implementation the most general possible, we consider the case in which we have more than three images.
First, we have to collect all the images, then we build the Intensity vector, that contains the intensities of a specific pixel in all the acquired images. Given the Light matrix previously computed, we compute its pseudo-inverse according to the Moore-Penrose operator. Then we are ready to compute the albedo (that is the norm of the product between the pseudo-inverse and the Intensity vector) and the normal for each pixel.

Algorithm 2 Photometry pseudocode

```
1: Load the Light matrix;
2: Compute its pseudo-inverse $LM_{pseudo}$;
3: Load all acquired images as single-channel matrices;
4: for all pixels in an image do
5:     Create a vector $I$ containing intensities of each image at a specific location $(h, w)$;
6:     Compute the product between $LM_{pseudo}$ and the Intensity vector;
7:     Compute the norm of the resulting vector to get Albedo at a specific pixel;
8:     if Albedo $\neq$ null then
9:         Compute the normal vector as the ratio between the resulting vector and albedo;
10:    else
11:        Set normal vector as null;
12:    end if
13:    Set the pixels for normal and albedo maps at location $(h, w)$;
14: end for
15: Save albedo and normal maps as images.
```

3.2 GPU implementation

Photometric stereo usually needs a large amount of data to get good results, and large amount of data requires a certain calculation power. To increase the speed of this process, we implemented Woodham's algorithm to make it work on a GPU [21].
Graphical Processing Units, or GPUs, are a particular type of computing processors used in modern computers mainly to render graphical images. Their architecture is highly suitable to manage and perform operations with matrices and vectors. Since 2001, when floating-point arithmetic started being supported, GPUs were used also to perform mathematical operations, not directly related with graphic processes. The scientific computing community's experiments with the new hardware began with a matrix multiplication routine; later, one of the first common scientific programs to run faster on GPUs than CPUs was an implementation of LU factorization. This usage of GPUs is usually referred to as "General Purpose computing on GPU" or GPGPU. Nowadays, there are several programming interfaces for graphics processors, as OpenGL, OpenCL and DirectX, that allow a customized usage of graphic processors. One of these interfaces is CUDA.
CUDA is a parallel computing platform and application programming interface (API) model created by Nvidia. The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements. It allows a fast parallelization, useful to solve large linear problems and to perform image processing.
CUDA code is executed as a sequence of functions, said *kernels*, that contain a set of instructions. Each kernel is synchronized with each other. For each kernel there is a determined number of threads, that are organized in blocks inside a grid. Grids can be at most three-dimensional structures.
The basic workflow is the following: first we have to upload the reference and result images to the GPU, then create a grid with a certain number of blocks. Each block will perform the instructions contained in the photometric stereo kernel on a limited set of pixels of the reference images.

In simple words, it is like splitting the images and send each part to a different block; after the operation, the resulting images are gathered together and downloaded to the CPU.

Algorithm 3 Photometry on GPU pseudocode

```
1: Load the Light matrix;
2: Compute its pseudo-inverse $LM_{pseudo}$;
3: Load all acquired images as single-channel matrices;
4: Create local destination matrices;
5: Upload acquired images, destination matrices and $LM_{pseudo}$ to GPU;
6: Create a GPU grid with a proper size, according to image height and
   width;
7: Start the GPU kernels, and wait until the device ends and synchronizes;
8: Download the results from GPU ;
9: Save results.
```

Chapter 4

Image acquisition system

In this chapter we will discuss about the hardware components that let us acquire the images. We will also briefly describe the control of the lightning system, required to synchronize it to the frame acquisition. In the final section there is an overview about the graphical user interface we developed.

4.1 Camera, light source and controller

The hardware is composed by a camera, a ring-light as a source and a controller.
The camera is an Allied Manta G-201B, that is a 2 Megapixel monochrome camera. It captures at a maximum resolution of 1624×1234 pixels, and it supports Ethernet connection with a maximum frame-rate of 14.7 fps. This camera is supported by a proprietary set of API (PvAPI) that can be employed through OpenCV.
As for the illuminator, we chose a CSS HPR2-150SW, that is a 4-sectors controllable light ring, that requires an input voltage of 24 V DC. It has an outer diameter of 166 mm and inner of 116 mm. As shown in Figure 4.1, the constructor guarantees an almost uniform illumination at different distances from the target.

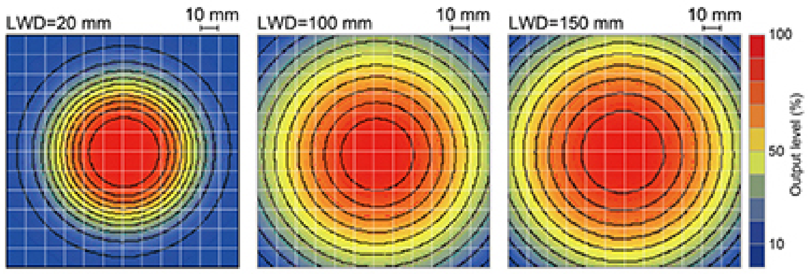


Figure 4.1: Uniformity of relative irradiance of the light ring CSS HPR2-150SW. LWD is the Light to Workpiece Distance.

Keeping the light ring at a distance of $150mm$ from the working object and using a $12mm$ lens on the camera, we can frame a region of about $14400mm^2$.
The controller is a Gardasoft RTCC420-20, that provides an Ethernet connection with TCP/UDP protocols. It has 4 output channels that can be used to control different light sources, and the constructor provides a SDK that lets us actively control the outputs.
The final hardware is shown in Figure 4.2.

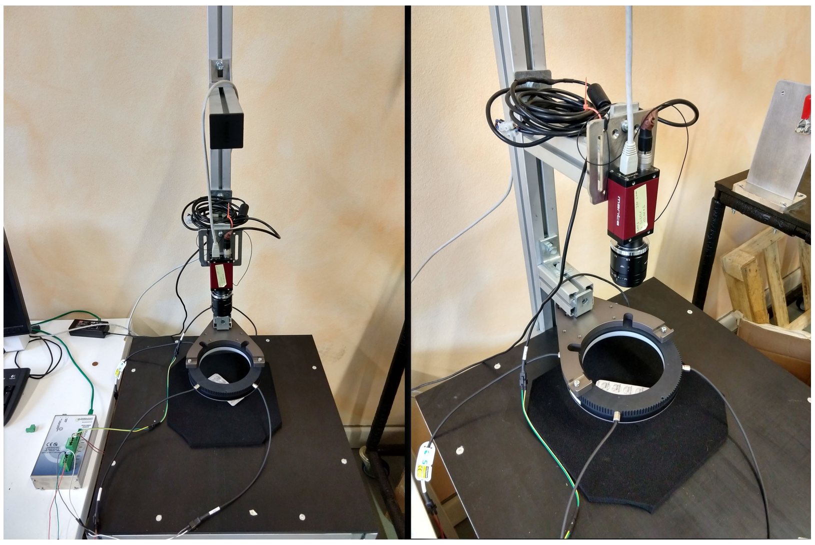


Figure 4.2: Two different views of our rig.

4.2 Image acquisition

One of the main concerns we encountered is the different time response of the devices. In fact, there was a latency between the input signal sent to turn on a light and the effective power on. In a first instance, this affected the image acquisition because the delay was not constant, so, even if a set of images was illuminated as expected, the subsequent one might have been wrong. The solution was simply to add a waiting time between the turn on of the light and the acquisition of the frame.

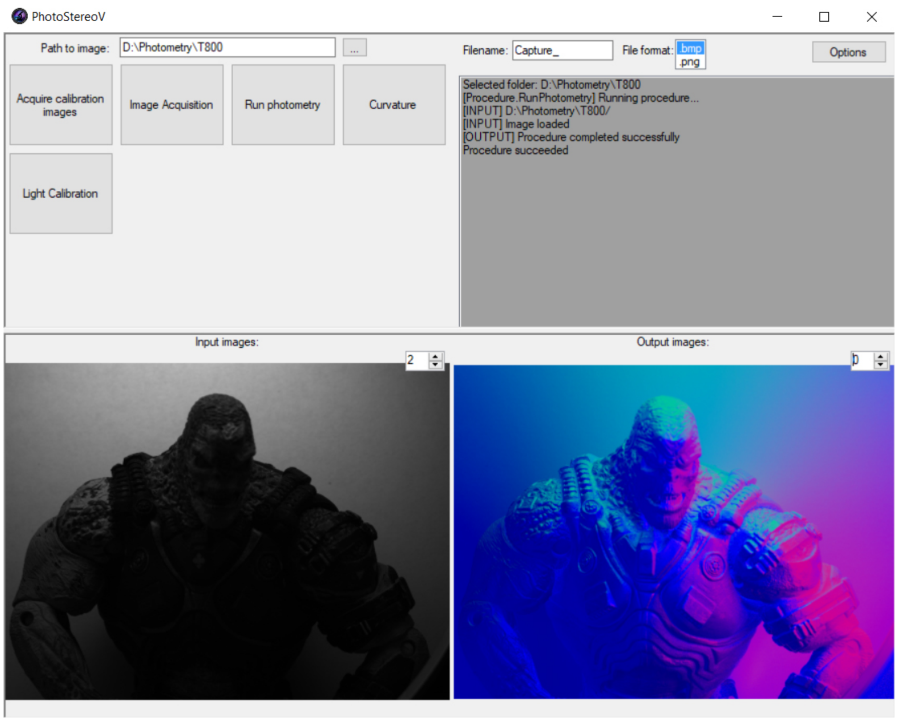
Algorithm 4 Image acquisition sequence

```
1: for each light channel do
2:     Send light turn-on command;
3:     Wait;
4:     Acquire image;
5:     Wait;
6:     Send light turn-off command.
7: end for
```

The command signals are strings that are sent to the controller via TCP, using the proprietary SDK. In each string, one must specify the channel to which we are sending the command, and the percentage of the light intensity. In fact, the power rating of the devices connected to the controller is made by the controller itself during the start-up.

4.3 Software User Interface

We developed a software with a graphical user interface, in C#, that lets the user easily perform different operations, from light calibration to photometric stereo, to curvature analysis. The software allows the user to specify the working folder and to see both input and output images. Moreover, it permits the choice of operating parameters, like the serial number of the controller, the exposure times for each image and the intensity of the light. In Figures 4.3 and 4.4 are shown two screens of our software.



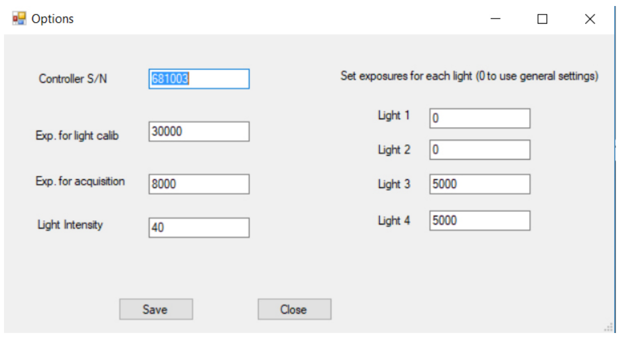
Path to image: D:\Photometry\T800

Filename: Capture_

File format: .bmp .png

```
Selected folder: D:\Photometry\T800
[Procedure.RunPhotometry] Running procedure...
[INPUT] D:\Photometry\T800/
[INPUT] Image loaded
[OUTPUT] Procedure completed successfully
Procedure succeeded
```

Figure 4.3: The main screen of the user interface



Controller S/N	681003
Exp. for light calib	30000
Exp. for acquisition	8000
Light Intensity	40

Set exposures for each light (0 to use general settings)

Light 1	0
Light 2	0
Light 3	5000
Light 4	5000

Figure 4.4: The options screen of our software

Chapter 5

Results

In this chapter, we will present the results we obtained with our photometric stereo system. In the first section we will show the results of the calibration process, and in the second there will be the results of the photometry algorithm, obtained using our acquisitions and some other reference images. Results will be also compared with the ones got with a popular commercial software, that is MVTec Halcon.

5.1 Light calibration

The calibration procedure has already been explained in Chapter 3, and we are briefly recalling it. We have to load the images, where a highly reflective black sphere has been lighted; an example is shown in Figure 5.1. Our algorithm requires then an opening, performed on all images, and then the recognition of the most reflective areas.

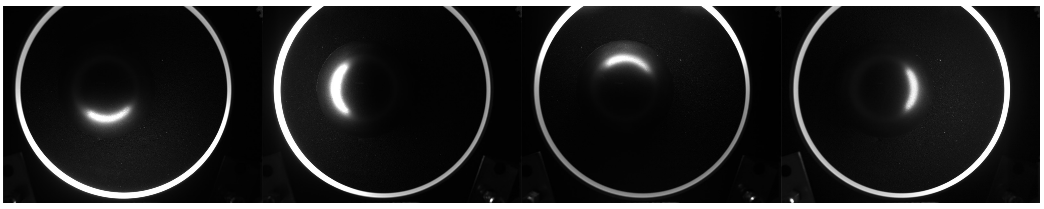


Figure 5.1: Four examples of a black sphere illuminated from different directions.

In Figure 5.2 it is shown an intermediate step, where there are the highly reflective spots and a line, that represents the planar distance from its center and the center of the sphere.

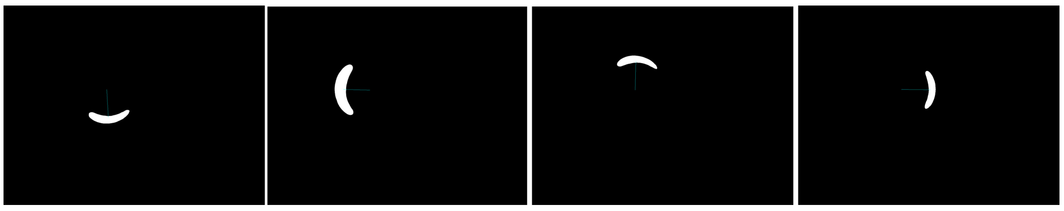
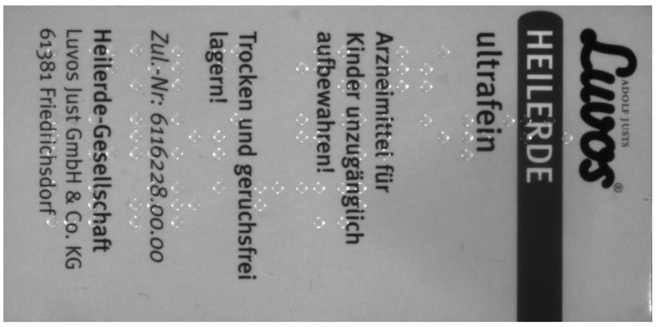


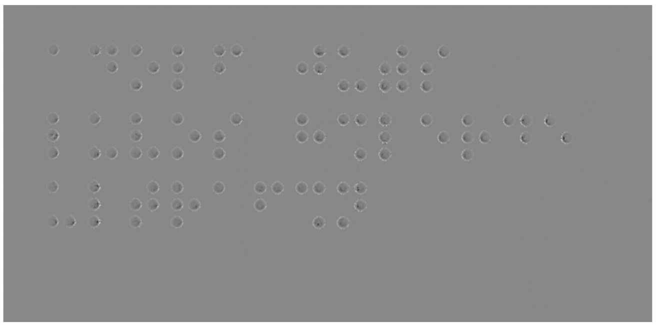
Figure 5.2: An intermediate step of the light calibration, showing the high reflective spots and, in blue, the segment connecting the center of the spot to the center of the sphere.

5.2 Photometry results

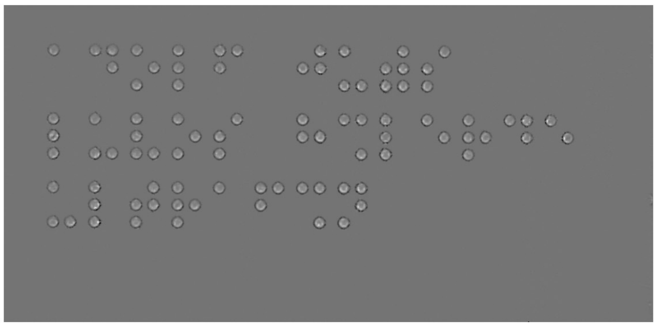
Photometric stereo can be used for several applications.
In Figure 5.3 there are the results of the process on a medicine package, where a braille text is impressed. Image A is the albedo, where it is practically impossible the recognition of braille characters, while in the two subsequent images there are the Gaussian curvatures obtained with our implementation (B) and with Halcon (C). Here we can notice the main difference, that is a post-processing that improves the contrast of Halcon's result, that makes the image more human-readable.
Another example is shown in Figure 5.4 that represents the photometric analysis performed on a marble tile. There are small defects, that are undetectable on albedo (A) but they are more visible using the mean curvature, as shown in images B and C, obtained - again - with our implementation and Halcon respectively.



A



B



C

Figure 5.3: Photometric results on a medicine package. A is albedo, B is the Gaussian curvature (with our software), C is the Gaussian curvature (with Halcon).

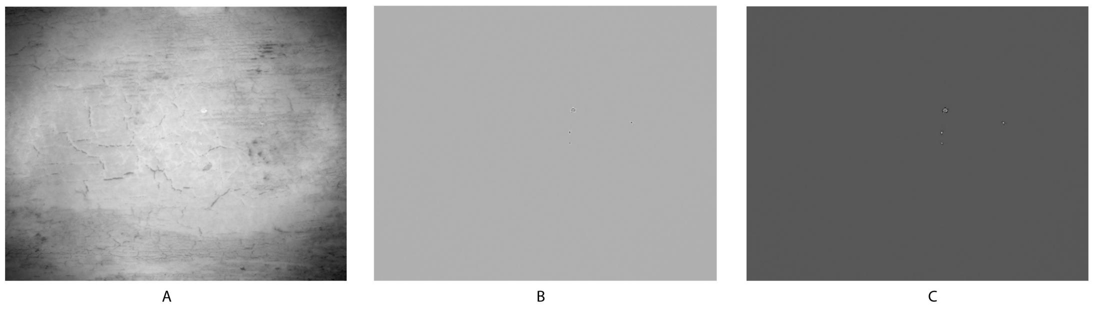


Figure 5.4: Photometric results on a marble tile. A is albedo, B and C are the mean curvatures with our software and Halcon respectively.

In Figure 5.5 there are the results on a small plastic piece where an identification number has been embossed. In image A it is more difficult to perform the character recognition (OCR) than using images B or D. Image C shows the normal map obtained with our software.

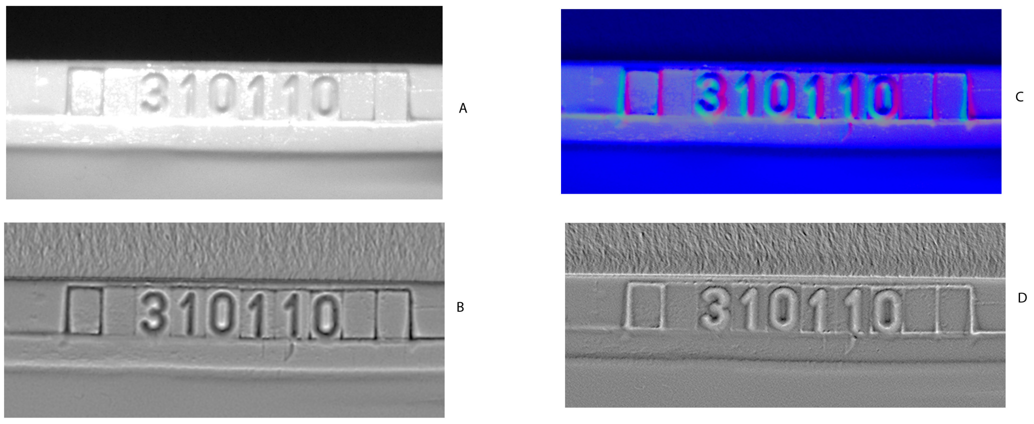


Figure 5.5: Photometric results on an embossed plastic piece. A is albedo, B is Halcon's mean curvature, C is the normal map and D is the mean curvature with our software.

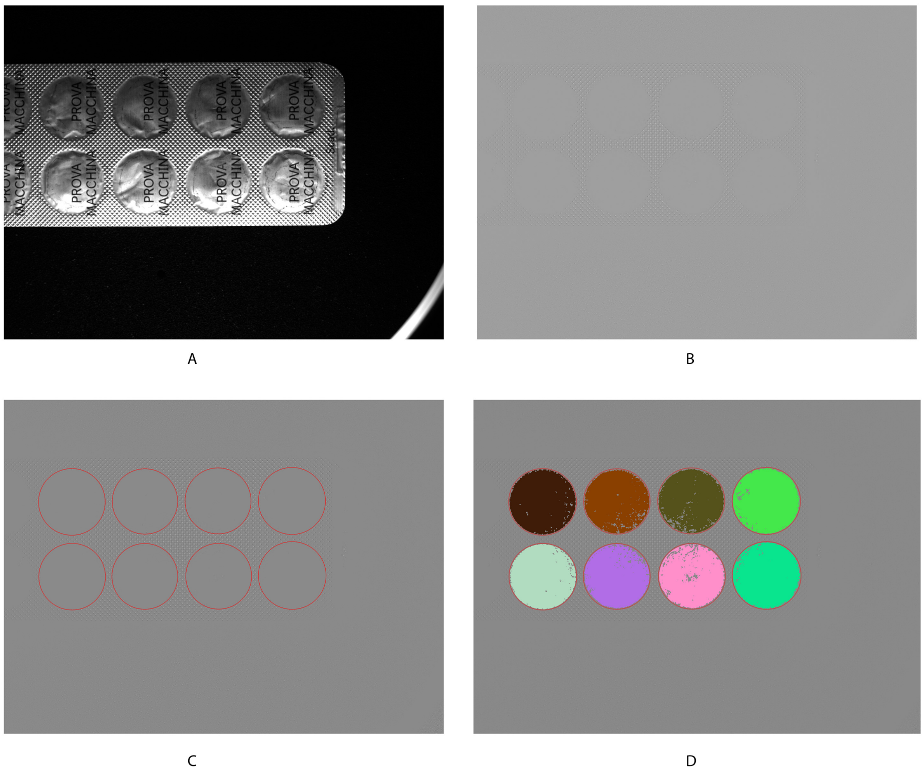


Figure 5.6: Photometric result and analysis on a blister pack. Image A is one of the acquired images, B is the Gaussian curvature. C is the result of a blob detection, and D is the result of a floodfill algorithm for labeling.

Figure 5.6 shows the results on a blister pack, whose images were acquired using our rig. Image A represents one of the acquired images, while B is the Gaussian curvature, that is almost unintellegible. However, we can apply some algorithms to detect the presence of blobs (shown in image C) after a thresholding and an edge detection, and then perform a labeling (image D). Another interesting result is shown in Figure 5.7, that is the mean curvature of a STM32 microcontroller. Further analysis may enhance details of the printed circuitry and of possible defects.

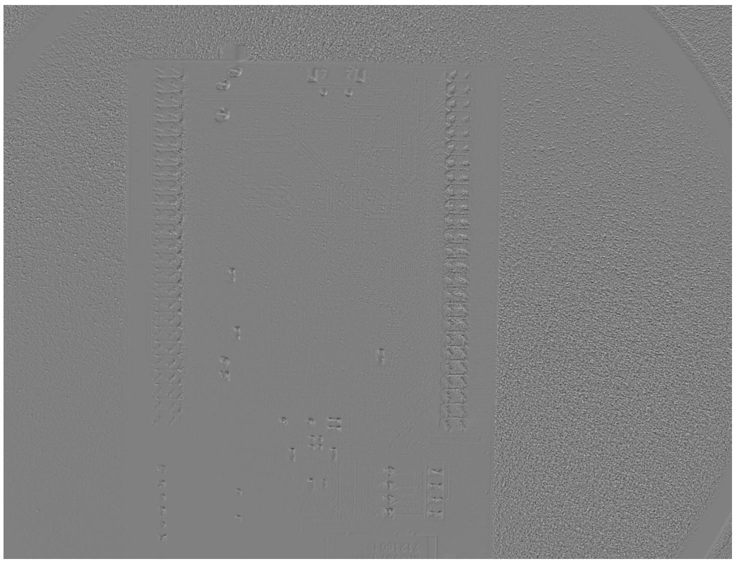


Figure 5.7: Mean curvature of a STM32 microcontroller.

In Figure 5.8 there are the results obtained on a small husky model. Image A is albedo, B is the normal map and C is the mean curvature. All the three have been computed with our software. Image D is the depth map obtained with the Matlab code, while E is the reconstructed height map obtained with Halcon, using a Poisson integrator.

Those height maps have been employed to get approximate 3D reconstructions. Results, obtained using a CAD software, are shown in Figure 5.9. Image A is the reconstruction based on the Matlab depth map, image B is based on Halcon's result. Both reconstructions are not as accurate as expected, but clearly image A is more reliable. In fact, even if the model used has not a strictly lambertian material, and its colors are not uniform, the reconstruction still maintains the most of the shape. This does not happen in image B, where a deformation is more evident.

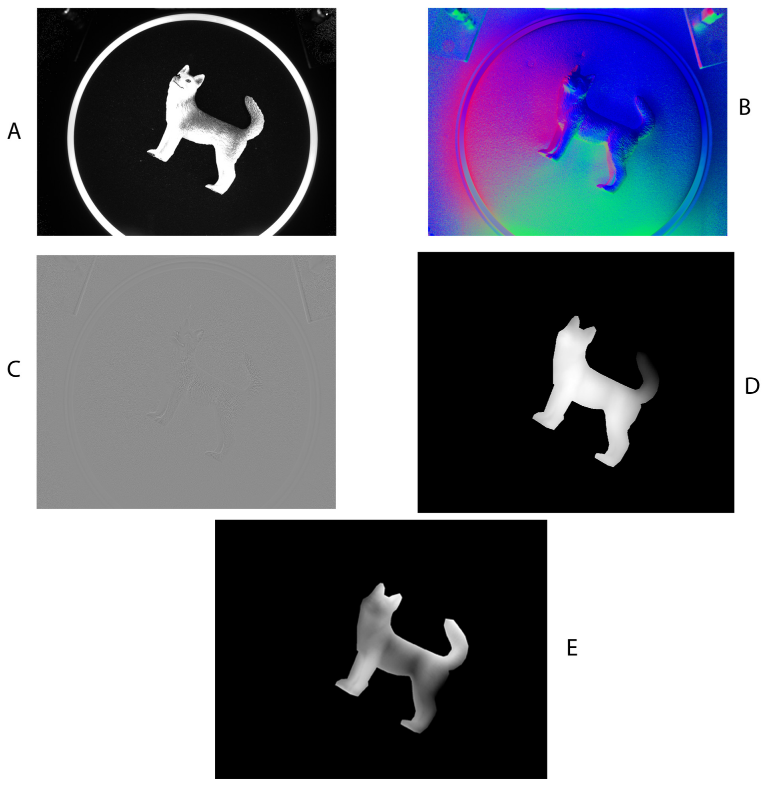


Figure 5.8: Results of photometry on a husky model. A is the albedo, B is the normal map, C is the mean curvature, D and E are height maps obtained with Matlab and Halcon respectively.

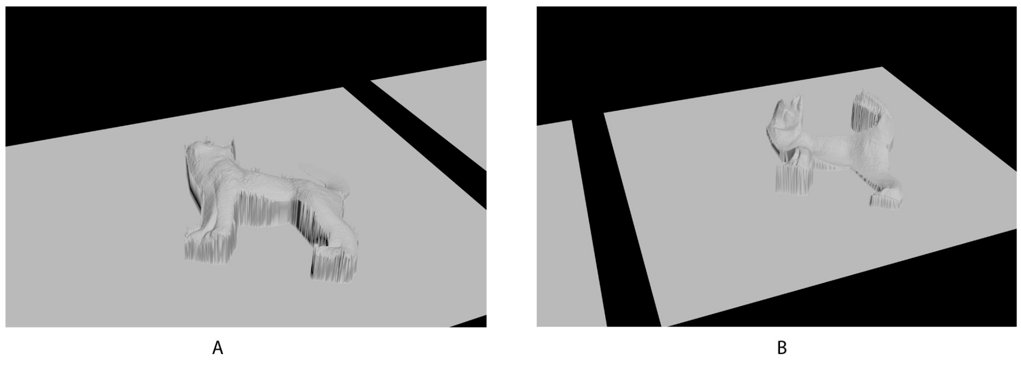


Figure 5.9: 3D reconstructions of the husky model obtained from Matlab's depth map (A) and Halcon (B).

Chapter 6

Conclusions

The goals of this thesis were the description of the main algorithms used to achieve a photometric stereo analysis of surfaces, as well as the realization of a hardware device capable of acquire the images, and of a software capable to analyze them. In particular, we focused on the implementation of Woodham's algorithm, and other image processing operators.
We briefly showed the most recent ways used nowadays to get a 3D reconstruction. We explained Woodham's algorithm, showing its simplicity and accuracy, but also its limitations. We described its analytical derivation, and the mathematical procedures used to calibrate the lighting system and get the height map.
In this thesis, we also showed how these algorithms were implemented, and showed the most significant results, that evidenced how, as for photometric stereo, our software and a commercial one are comparable.
However, further improvements are possible.
In fact, both camera resolution and light system can be upgraded to get more precise results. Moreover, the depth map reconstruction algorithm can be improved, in terms of speed and final reliability, and can be implemented to be processed directly on a GPU.

Bibliography

[1] E. Prados and O. Faugeras, ”Shape from shading: a well-posed problem?”, Project Odyssée, Report 5297, 2004

[2] R.J. Woodham, ”Photometric method for determining surface orientation from multiple images”,Optical Engineering Vol 19 n.1, 1980

[3] A.D. Jepson and D.J. Fleet, ”Lighting and reflectance models”, 2011

[4] H. Hayakawa, ”Photometric stereo under a light source with arbitrary motion”, J.Opt.Soc.Am. A/Vol 11, 1994

[5] B.K.P. Horn, ”Height and gradient from shading”,MIT AI Lab, A.I. Memo No. 1105A, 1990

[6] R.T. Frankot and R. Chellappa, ”A method for enforcing integrability in shape-from-shading algorithms”, IEEE transactions on pattern analysis and machine intelligence, Vol 10, 1988

[7] Gary A. Atkinson, Lecture about ”Photometric Stereo in 3D Biometrics”, 2010

[8] Hao Li, CSCI 621: Digital Geometry Processing, 2017

[9] Nayar, Krishnan, Grossberg, Raskar, ”Fast Separation of Direct and Global Components of a Scene using High Frequency Illumination”, Cambridge, 2006

[10] J. Wang et al., ”Gigapixel 3D Camera”, 2016

[11] P. Tan et al., ”Resolution-Enhanced Photometric Stereo”, Springer-Verlag Berlin Heidelberg,2006

[12] A. Jones et al., ”Head-mounted Photometric Stereo for Performance Capture”, University of Southern California Institute for Creative Technologies

[13] D. Vlasic et al., ”Dynamic Shape Capture using Multi-View Photometric Stereo” MIT

[14] L. MacDonald et al., ”Accuracy of 3D reconstruction in an illumination dome”, The International Archives of the Photogrammetry, 2016

[15] Basri, Jacobs and Kemelmacher, ”Photometric Stereo with General, Unknown Lighting”, International Journal of Computer Vision, Springer 2006

[16] Robin Berntsson, ”Photometric stereo for eye tracking imagery”, Independent thesis Advanced level , 2017

[17] MVTec Software GmbH, 2018, HALCON Operator Reference 13.0.2

[18] P.Grinfeld, ”Introduction to Tensor Analysis and the Calculus of Moving Surfaces”, 2014, Springer

[19] OpenCV website, at https://opencv.org , 2018, OpenCV team

[20] Xiuming Zhang, at https://github.com/xiumingzhang/photometric-stereo, 2015

[21] Luc Van Gool, Leuven, Pevar, Verswyvel et al., ”Real-time photometric stereo”, Wichmann/VDE Verlag, Belin & Offenbach, 2015

Acknowledgements

I would like to express my gratitude to my tutor, eng. Luca Piccinini, and professor Luigi Di Stefano for their great experience and guidance.
My sincere thanks also go to all the T3Lab guys and university colleagues, that have been wonderful journey companions.
My best thanks go to my parents, my brothers and my friends, who gave me support, and who believed in me.