Advances

in

Photometric Stereo

Ruggero Pintus

Tutor – Prof. Massimo Vanzi

Contents

1. Introduction

Photometric Stereo is a fundamental part of the entire process concerning computer vision [1]. This issue is very important in industrial applications, artificial intelligence researches and so on. It's based on an image processing procedure that analyses the scene and tries to recover its three-dimensional surface. There are a lot of techniques to 3D map a single object or an entire scene and each of them has different features regarding the approach to the problem. The most used is laser scanning techniques, since it's able to 3D map separately and, most of all, independently each single point of a sample. Instead, almost all other methods need to analyse a pixel cluster or even all pixels in N images to recover surface heights. Obviously this represents a drawback, since the height value of one point (whose data is supposed noiseless) will be biased by the noise in the adjacent pixels. For this reason all of these methods are carefully used and, in most cases, are merged with other more accurate methods [2]. However, depending on the nature of samples and image acquisition systems, sometimes we are forced to make use of only one technique. For instance in section 7 we show the 3D map of object observed with scanning electron microscope (SEM); we believe that it's really difficult to use a laser scanning inside the SEM, don't you?

In this sense we have focused our efforts in solving some trouble concerning both the management of non-idealities (with respect to the mathematical model) and the numerical algorithm, to enhance its performances in terms of accuracy and computational time. Before showing our work we want to list some reference about previous works in Photometric Stereo and generally in Shape from X (e.g. shape from shading) algorithms.

Photometric Stereo was conceived by Woodham [3] that takes the theory of shape from shading, which reconstructs the heights using only one image, and extends it to a number $N \geq 2$ images. This technique uses a reflectance map as a mathematical model or a lookup table, which decreases the computational time in some simple applications. The lookup tables are built with a calibration sphere of known shape and material behaviour. Depending on used algorithm the sphere should be of the same material of objects to be 3D mapped. On the other hand, the algorithms that reconstruct also albedo parameters don't need this constraint. Note that, if there is no need to find albedo it's better to acquire only gray-scale images, to minimize used memory, while colour images are obviously necessary if we need chromatic information. Generally the vast majority of methods use gray-scale images. For Lambertian surface, as we'll see in next sections, photometric equations are linear and only three images are enough to reconstruct both gradient and albedo. With more than three images some papers use the "surplus" information to find other object part, as outliers [4], or to recover unknown illumination directions and strengths [5]. While some methods start from a given reflectance map and then find the heights, other techniques focus their attention to estimate reflectance parameters.

Some instances are Nayar [6], who uses the so-called hybrid reflectance model, and Tagare and de Figueiredo [7], who developed a photometric stereo for the class of m-lobe reflectance maps (they called it so). As we'll also see in this work, many papers deal with the possibility to consider a surface as Lambertian even if it has non-idealities as shadows or highlights. The trick is to consider a mathematical model that define highlights and shadows as deviations from Lambertian law; in this way the algorithm can manage them and reconstruct surfaces properly. One of these techniques was proposed by Coleman and Jain [8].

On the other hand, colour images contain a lot of redundant information regards the gradient extraction. If one could have this data, he should use this information whether to make the algorithm more robust or to compute more than heights. Christensen and Shapiro [9] introduced the method of colour photometric stereo (CPS) for surfaces with an arbitrary reflectance map. A drawback of this technique is that it needs a calibration step, while (other papers already do that) we'll demonstrate in the following sections that it's not necessary. Other papers (as [10]) separate the Lambertian signal to the others with an image processing tool, which permits to avoid any non-ideality bias.

Before starting to face our issue we list here the sections of this work. Section 1 describes the basics in photometric stereo physical model and finds the most used reflectance map equation for Lambertian surfaces. Section 2 shows the first steps we had done studying PS and especially the changing from using two to four light sources and lays the bases of next sections. Then, Section 3 illustrates the theory of gradient and albedo extraction. Section 4 deals with an algorithm we have developed to avoid illumination biases. In Section 5 we explain our gradient integration algorithm and compare it with existing ones. We also study its performances in terms of computational time and robustness. Section 6 is made by all applications we are contended with and we'll show there our algorithm potential. The last two sections are the Appendix A, in which we demonstrate the mathematical details in choosing the best lighting conditions used to light the analytical PS sample (the vase), and references.

2. Image formation and reflectance map

In this section we deal with the image formation process and we find a mathematical model that relates object and light geometry with pixels intensity. First we introduce some notions as radiometry, irradiance, radiance, image brightness and bidirectional reflectance distribution function. The latter is the most important function that, together with known light sources position, leads to reflectance map formulation. Following theory sums up the basics of photometric stereo as discussed in [1].

To understand how the pixels intensity in acquired images is determined, we should study some radiometry concepts. First, the irradiance is the amount of light falling on a surface and it's the power per unit area incident on it (W/m^2). The radiance, instead, is the amount of light radiated from a surface and it's measured as the power radiated from the surface per unit area per unit solid angle ($W/m^2 sr$). Remember that the solid angle is defined as:

$$\Omega = \frac{A\cos\vartheta}{R^2} \qquad (2.1)$$

where A is surface area, ϑ is the angle between surface normal and a line connecting the patch to the origin and R is the length of this line.

Starting from these definitions we now discuss the image formation process. Let's consider Fig. 2.1, where there are one object, a lens and an image plane.

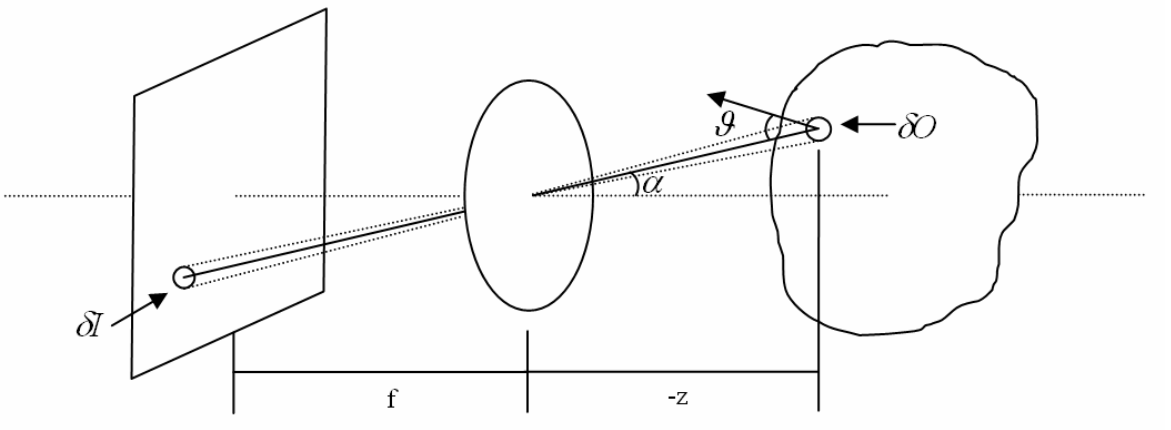


Fig. 2.1 – Image formation system

As easy to see, if the rays passing through the lens are not deflected, the solid angle concerning the object is equal to the solid angle regarding the image plane. These solid angles are:

$$\begin{cases} \dfrac{\delta I \cos\alpha}{\left(f / \cos\alpha \right)^2} \\ \dfrac{\delta O \cos\vartheta}{\left(z / \cos\alpha \right)^2} \end{cases} \qquad (2.2)$$

and if they are equal, it holds

$$\frac{\delta O}{\delta I} = \frac{\cos\alpha}{\cos\vartheta}\left(\frac{f}{z} \right)^2 \qquad (2.3)$$

Now, we should investigate how much of the light emitted from the surface patch go through the lens. The solid angle subtended by the lens, as seen from surface patch is

$$\Omega = \frac{\pi}{4}\frac{d^2 \cos\alpha}{\left(z / \cos\alpha \right)^2} = \frac{\pi}{4}\left(\frac{d}{z} \right)^2 \cos^3\alpha \qquad (2.4)$$

where d is diameter lens.

The power of the light emitted from the surface patch and falling on image plane is

$$\delta P = L\Omega\delta O\cos\vartheta = L\delta O\frac{\pi}{4}\left(\frac{d}{z} \right)^2 \cos^3\alpha\cos\vartheta \qquad (2.5)$$

where L is the radiance of the surface in the direction toward lens. The irradiance of the image plane patch will be

$$E = \frac{\delta P}{\delta I} = L\frac{\delta O}{\delta I}\frac{\pi}{4}\left(\frac{d}{z} \right)^2 \cos^3\alpha\cos\vartheta \qquad (2.6)$$

Using (2.3) we find this relationship

$$E = L\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4 \alpha \qquad (2.7)$$

Thus image irradiance is proportional to scene radiance L and to the angle alpha between the ray connecting the center of the lens and the patch and the optical axis. This is the first important result in the procedure that leads to reflectance map.

We saw how image irradiance depends on scene radiance, but what does radiance depend on? It depends on the light that falls on the object and the fraction of this light that is reflected, due to many reasons as object geometry or object reflection properties. The equation that describe what we have just said is called bidirectional reflectance distribution function (BRDF)

$$f(\vartheta_i, \varphi_i; \vartheta_e, \varphi_e) = \frac{\delta L(\vartheta_e, \varphi_e)}{\delta E(\vartheta_i, \varphi_i)} \qquad (2.8)$$

where (ϑ_e, φ_e) and (ϑ_i, φ_i) are the viewing and lighting direction (Fig.2.2).

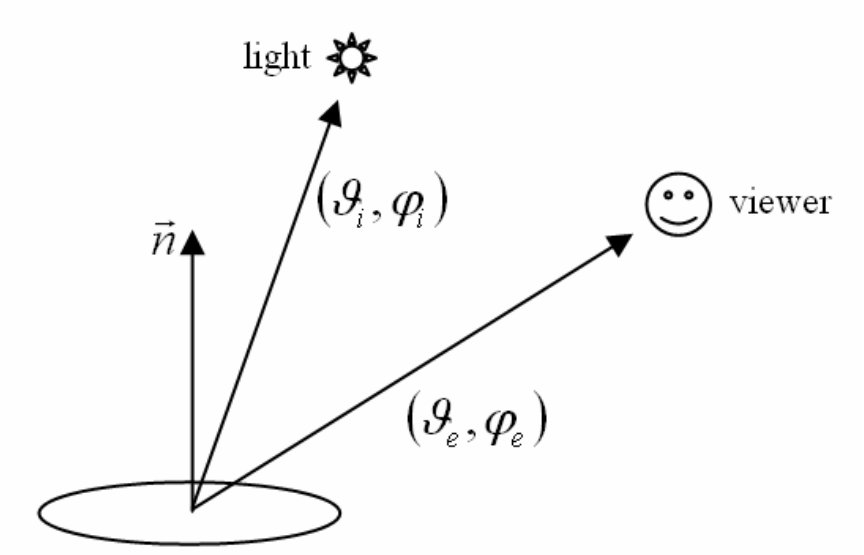


Fig. 2.2 – BRDF geometry

One important property of this function consists in Helmholtz reciprocity condition

$$f(\vartheta_i, \varphi_i; \vartheta_e, \varphi_e) = f(\vartheta_e, \varphi_e; \vartheta_i, \varphi_i) \qquad (2.9)$$

Generally we don't have a single specific direction which the light comes from, but we have a "sky" with different radiance that depends on the sky region. Consider such a sky as a hemisphere and a part of it of size $(\delta\vartheta_i, \delta\varphi_i)$. Supposed that a patch is in the hemisphere center and then it subtends a solid angle

$$\delta\omega = \sin\vartheta_i \delta\vartheta_i \delta\varphi_i \qquad (2.10)$$

The irradiance received from this part of the "sky" will be

$$E(\vartheta_i, \varphi_i)\sin\vartheta_i \delta\vartheta_i \delta\varphi_i \qquad (2.11)$$

So, the total irradiance is

$$E_0 = \int_{-\pi}^{\pi} \int_{0}^{\pi/2} E(\vartheta_i, \varphi_i)\sin\vartheta_i \cos\vartheta_i \delta\vartheta_i \delta\varphi_i \qquad (2.12)$$

where $\cos\vartheta_i$ accounts for the foreshortening of surface seen from the direction (ϑ_i, φ_i).

So, the radiance of the surface will be

$$L(\vartheta_e, \varphi_e) = \int_{-\pi}^{\pi} \int_{0}^{\pi/2} f(\vartheta_i, \varphi_i; \vartheta_e, \varphi_e) E(\vartheta_i, \varphi_i)\sin\vartheta_i \cos\vartheta_i \delta\vartheta_i \delta\varphi_i \qquad (2.13)$$

The same approach has to be considered if we want to compute the radiance of a Lambertian surface. In an ideal Lambertian surface one patch appears equally bright from all viewing directions and doesn't absorb any radiation power. So, the BRDF function must be constant. To find that value we have to use the previous integrals, setting that the total irradiance is equal to radiance integral over all directions, so

$$\int_{-\pi}^{\pi}\int_{0}^{\pi/2} f(\vartheta_i,\varphi_i;\vartheta_e,\varphi_e)E\cos\vartheta_i\sin\vartheta_e\cos\vartheta_e\delta\vartheta_e\delta\varphi_e = E\cos\vartheta_i \qquad (2.14)$$

which becomes

$$2\pi f\int_{0}^{\pi/2}\sin\vartheta_e\cos\vartheta_e\delta\vartheta_e = 1 \qquad (2.15)$$

This leads to

$$f = \frac{1}{\pi} \qquad (2.16)$$

So,

$$L = \frac{1}{\pi}E_0 = \frac{1}{\pi}E\cos\vartheta_i \qquad (2.17)$$

We have found the relation that correlates the direction and intensity of illumination with the surface radiance, which the pixels brightness depends on. Now we have only to write this relation in terms of surface derivatives along x and y.

We can easily demonstrate that a normal vector could be written as

$$\vec{n} = \frac{(-p,-q,1)^T}{\sqrt{1+p^2+q^2}} \qquad (2.18)$$

In the same way and for using similar notation we could define the illumination vector as

$$\vec{l} = \frac{(-p_S, -q_S, 1)^T}{\sqrt{1 + p_S^2 + q_S^2}} \qquad (2.19)$$

Now, using these equations,

$$\cos \vartheta_i = \frac{1 + p_S p + q_S q}{\sqrt{1 + p^2 + q^2}\sqrt{1 + p_S^2 + q_S^2}} \qquad (2.20)$$

Since this equation is the unique term that really influence the pixels brightness, while the other terms are only scale factors, usually we define a normalized reflectance map $R(p,q)$

$$R(p,q) = \frac{1 + p_S p + q_S q}{\sqrt{1 + p^2 + q^2}\sqrt{1 + p_S^2 + q_S^2}} \qquad (2.21)$$

As we shall see in next section this kind of reflectance map doesn't care about the surface albedo. This surface property should be added as an important scale factor that depends on the nature of each single area patch corresponding with one image pixel.

Before moving on with the next section, Photometric Stereo basics are now described. Consider now the (p,q) space and an acquired image of a surface with uniform albedo. Let us consider also a single pixel with value E_1. Then we have

$$R(p,q) = E_1 \qquad (2.22)$$

This equation is a curve in the (p,q) plane. We can't compute the (p,q) value matching E_1, since there are infinite values of this gradient pair. We need much more information. We need more images with different light condition (different $\vec{l}$ vector)! Other images represent other curves and the intersection point of these curves is the gradient of the analysed point. Fig. 2.3 shows reflectance map curves for some E values, setting $p_S = 1$ and $q_S = 0.5$. This is the starting point for the development of all PS algorithms.

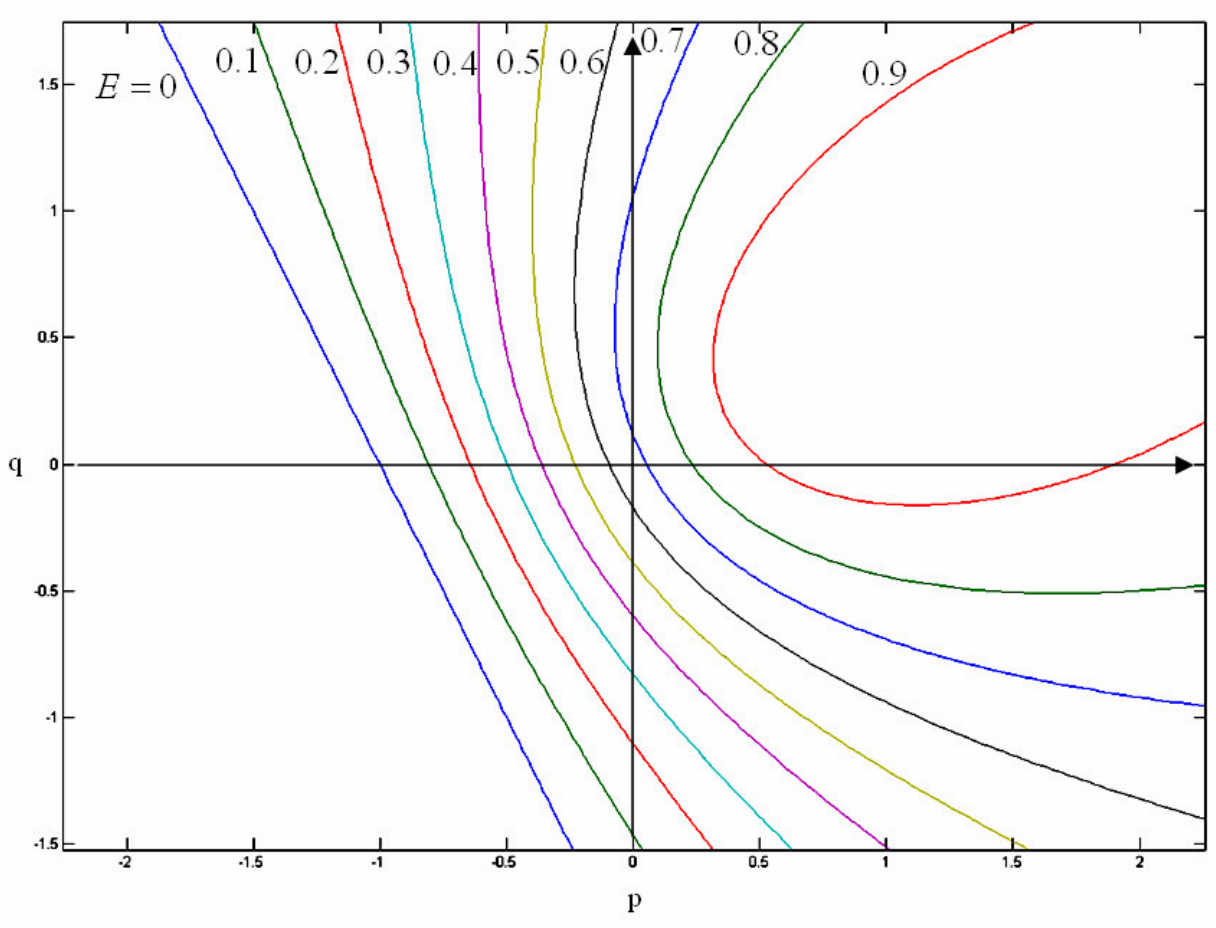


Fig. 2.3 – Reflectance map curves for ten values of E

3. Changing from two to four lights

When we started to apply PS algorithm to real surfaces, we used a particular form of the reflectance map seen in previous chapters to compute the gradient field and we placed only two lights, one on the right and the other on the left of the camera,

$$I_R = I_0 \rho(x,y) \cos\alpha \frac{1 - \tan\alpha \left[\frac{\partial z}{\partial x} \cos\beta + \frac{\partial z}{\partial y} sen\beta \right]}{\sqrt{1 + \left(\frac{\partial z}{\partial x} \right)^2 + \left(\frac{\partial z}{\partial y} \right)^2}} \qquad (3.1)$$

where I_R is the acquired object image, which depends on the scene geometry and on object reflectance properties, I_0 is the intensity of the light, $\rho(x,y)$ is the albedo and (α,β) defines the position of each light (Fig. 3.1).

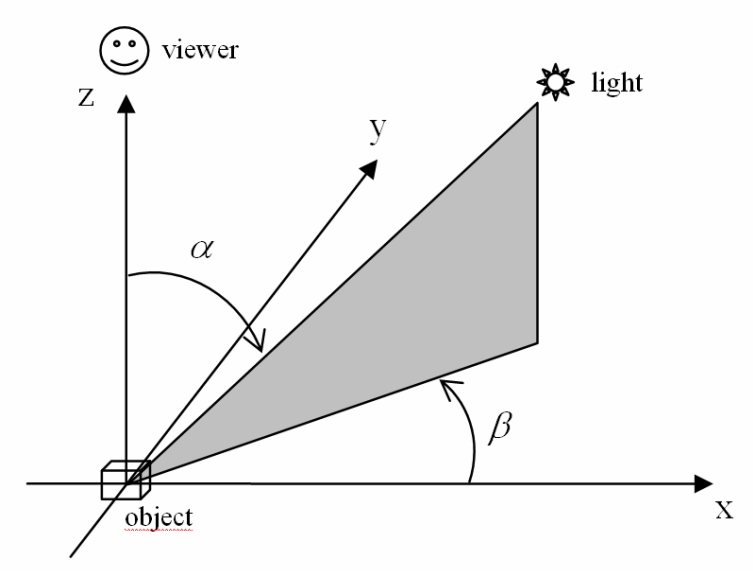


Fig. 3.1 – System geometry

With this model we used to compute the derivatives in such an easy way: in the case of derivative along x we consider $\beta_1 = 0$ and $\beta_2 = \pi$ and easily find

$$\begin{cases} I_{R1} = I_0 \rho(x,y)\cos\alpha \dfrac{1-\tan\alpha\dfrac{\partial z}{\partial x}}{\sqrt{1+\left(\dfrac{\partial z}{\partial x}\right)^2+\left(\dfrac{\partial z}{\partial y}\right)^2}} \\ I_{R2} = I_0 \rho(x,y)\cos\alpha \dfrac{1+\tan\alpha\dfrac{\partial z}{\partial x}}{\sqrt{1+\left(\dfrac{\partial z}{\partial x}\right)^2+\left(\dfrac{\partial z}{\partial y}\right)^2}} \end{cases} \qquad (3.2)$$

$$p = \frac{\partial z}{\partial x} = -\frac{1}{\tan\alpha}\frac{I_{R1}-I_{R2}}{I_{R1}+I_{R2}}$$

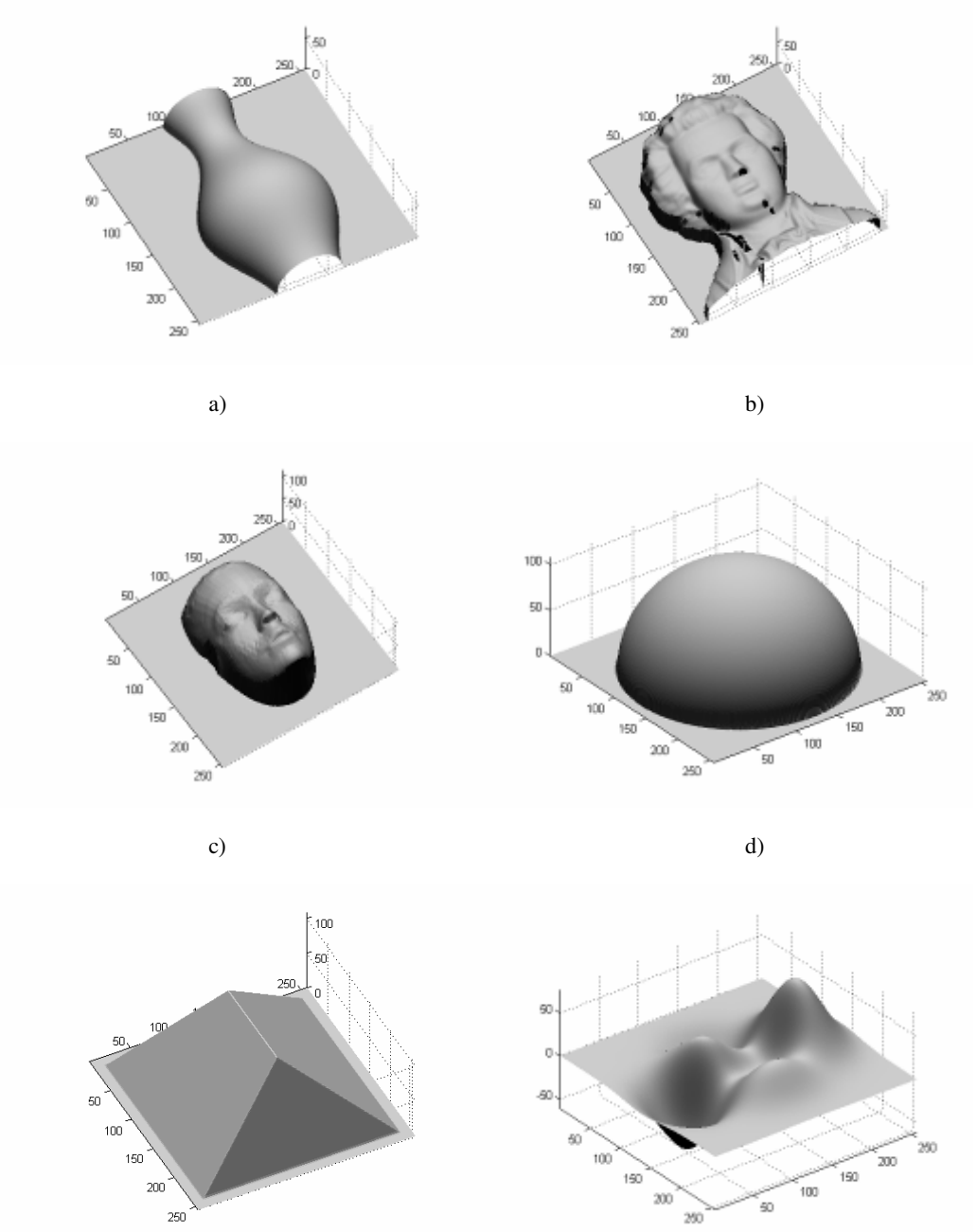
and in the same way, choosing $\beta_1 = \dfrac{\pi}{2}$ and $\beta_2 = -\dfrac{\pi}{2}$

$$q = \frac{\partial z}{\partial y} = -\frac{1}{\tan\alpha}\frac{I_{R1}-I_{R2}}{I_{R1}+I_{R2}} \qquad (3.3)$$

However, this leads to a big problem, since with two lights I could obtain only a part of the gradient field that is absolutely necessary for computing the depth map. With one derivative I could only know the height map of one row along x a time (or column along y) but I can't establish a connection with the height information of adjacent rows (or column). For this reason the first 3D reconstructions are very poor and there is a great problem with noise effects on them.

The first solution is to use the same formulas but four lights, the first two ones for the x derivative and the latter ones for the y derivative. It's obvious that this approach increases enormously the 3D reconstruction algorithm possibilities. Let's see an example of this first simple approach at the photometric stereo issue.

In every part of this thesis we'll use always one particular PS sample, which permits us to compare different methods, whether a gradient extraction or an integration algorithm is performed. Fig.3.3 shows this object to be 3D mapped. One big issue in photometric stereo and shape from shading algorithms is accuracy estimation, since, if a real sample is chosen, rarely it's possible to numerically compare the real geometry with the reconstructed one. So, in addition to a real sample, we have to consider, during this work, also an analytical one. Typically there are some PS and SFS numerical surface (Fig.3.2) [11]. We choose the vase and the reasons of this will be clear when the integration algorithms will be developed and analysed.



a) b)

c) d)

e) f)

Fig. 3.2 – typical PS and SFS test images

Fig. 3.3 – the PS sample in this work to test used algorithms

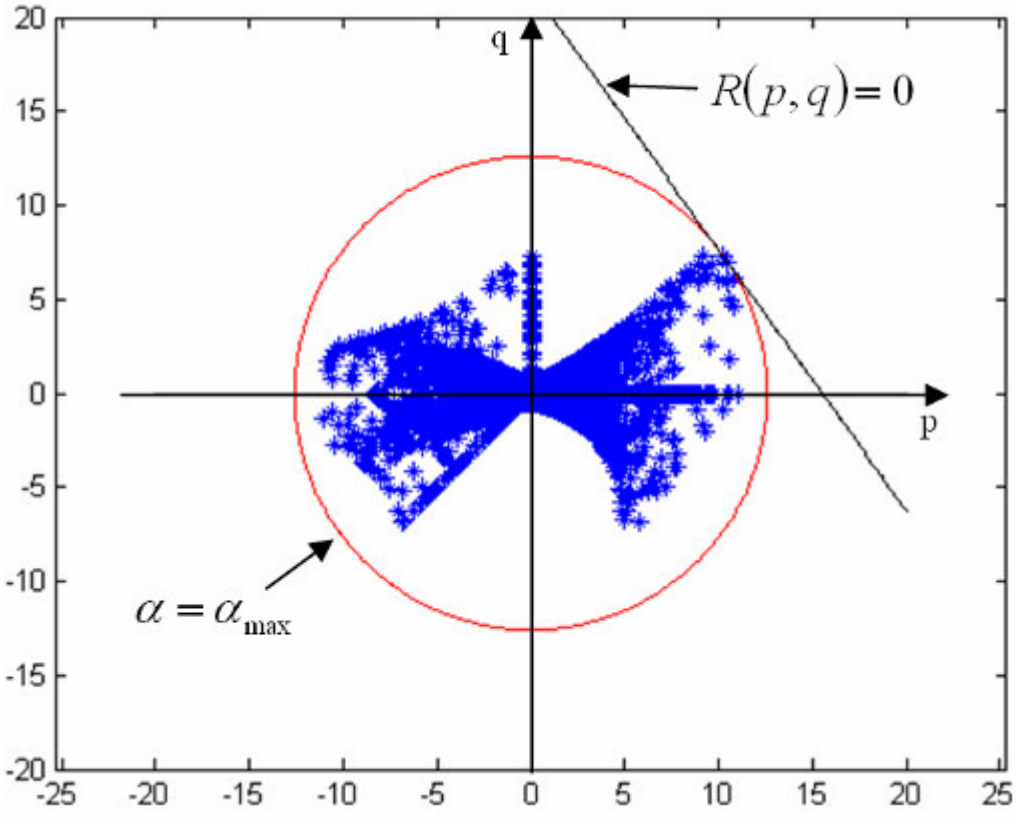


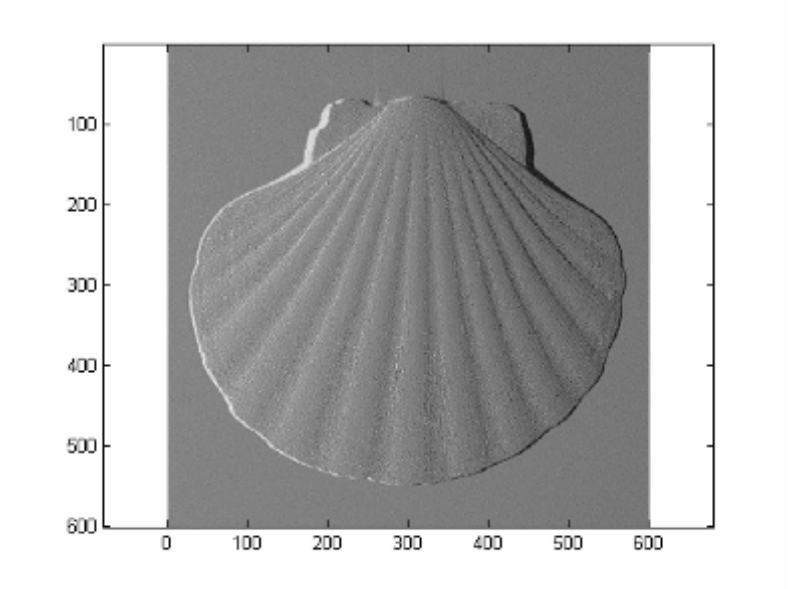
Fig. 3.4 – The circle with the minimum ray that includes all surface (p,q) pairs was displayed in red and it corresponds to the maximum alpha value that doesn't allow shadows in gray scale images

In the case of the vase we have to choose the best illumination condition. Referring to appendix A, we should analyse the gradient field in the (p,q) space to solve this task. Precisely, considering all possible beta values, we have to find the alpha value that doesn't in any case make the shadow line intersect the minimum circle that includes all surface points plotted in the (p,q) plane. Fig.3.4 shows vase gradient pairs plotted in (p,q) plane and the minimum circle that contains them.

The coordinates of the most distant (p,q) pair are $(10.2,7.4)$. Using this parameter in (8.9) in the appendix we could find that $\alpha_{\max} = 0.0792rad \approx 4.5°$.

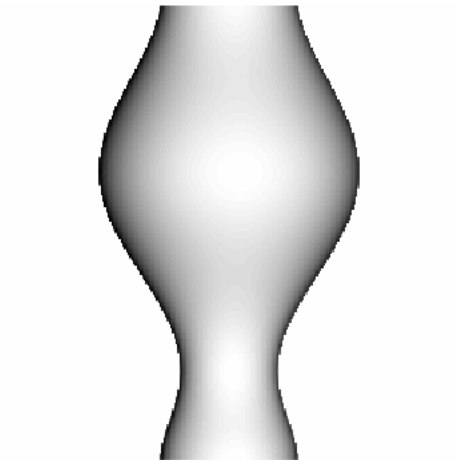
Using the equations above we could compute the derivatives along x and y of both shell and vase. Starting with the x one we have to consider $\beta_1 = 0$ and $\beta_2 = \pi$. Fig. 3.5 and 3.6 show (I_{R1}, I_{R2}) of shell and vase and their x derivative. Instead, for the y derivative we put $\beta_1 = \frac{\pi}{2}$ and $\beta_2 = -\frac{\pi}{2}$. Fig. 3.7 and 3.8 show (I_{R1}, I_{R2}) of shell and vase and their y derivative. Note that in shell case we don't know the numerical exact value of alpha (assuming parallel rays, see section 5), but it doesn't matter, because it will only affect scale 3D reconstruction.

a) b)

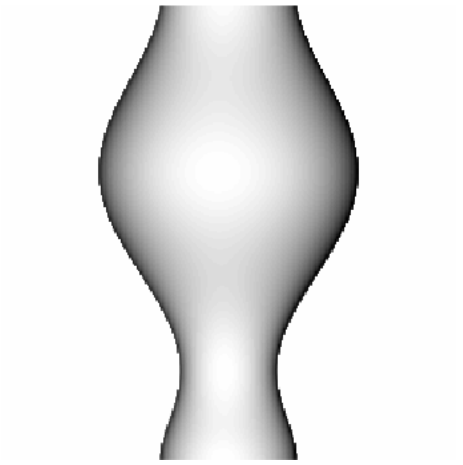


c)

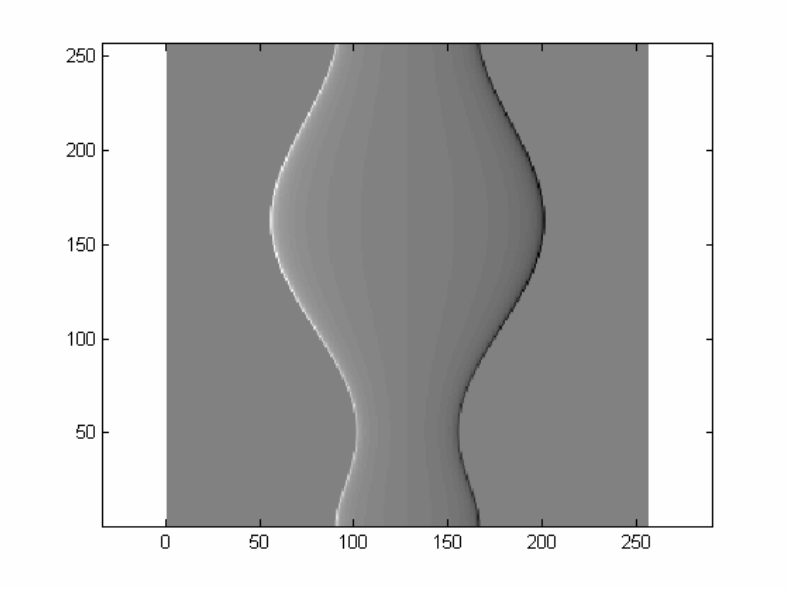
Fig. 3.5 – a) and b) show shell images lit respectively from right and left, while c) surface derivative along x



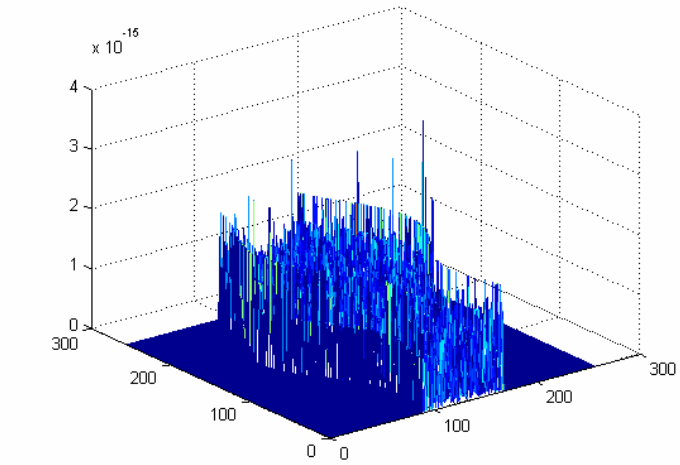
a)



b)



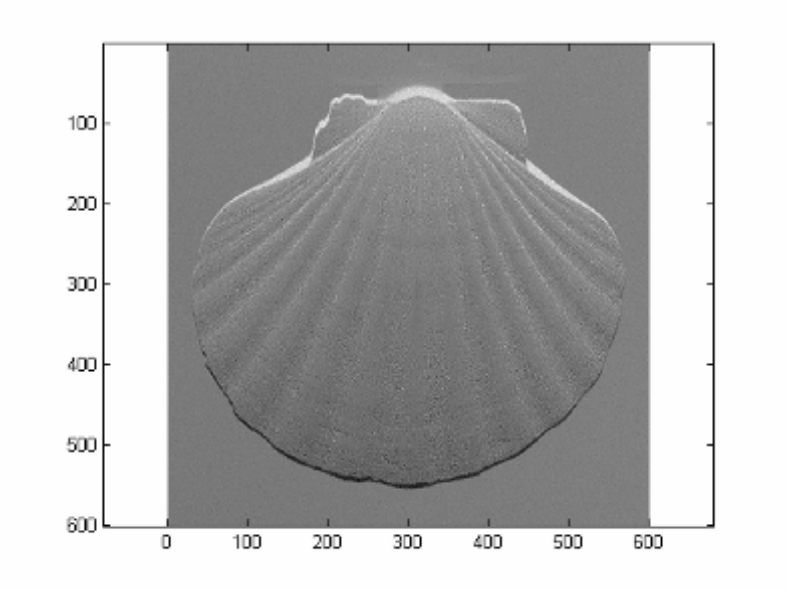
c)



d)

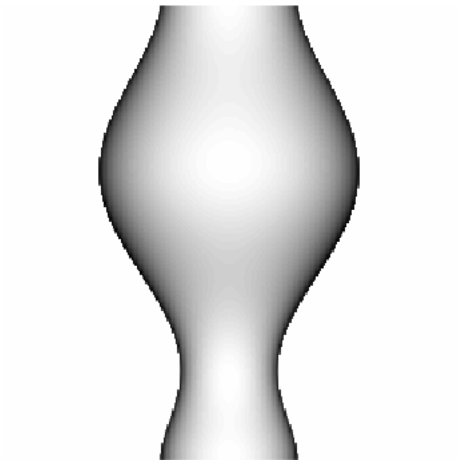
Fig. 3.6 – a) and b) show vase images lit respectively from right and left, while c) surface derivative along x. d) is the error between the original x derivative and the computed one

a) b)

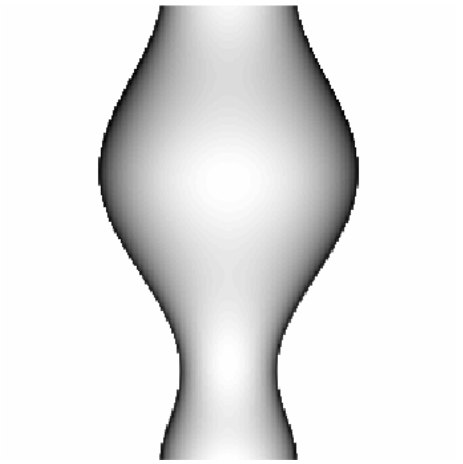


c)

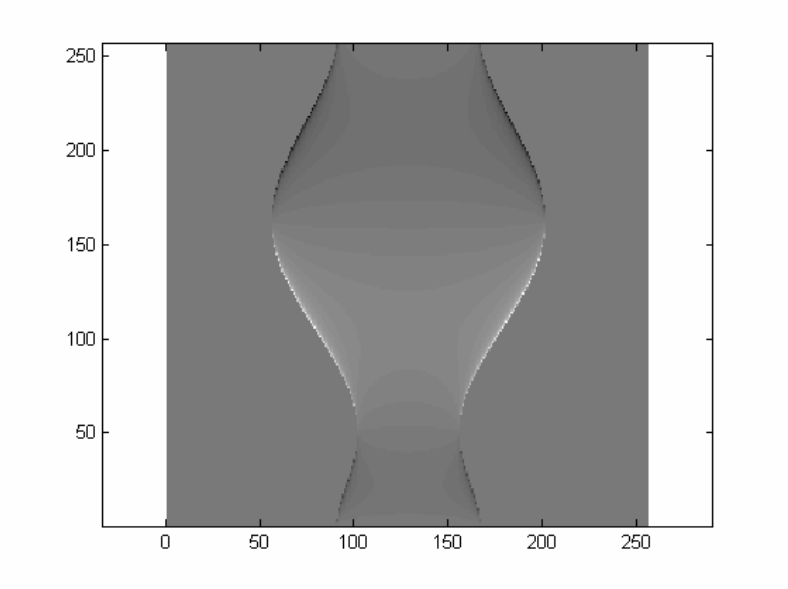
Fig. 3.7 – a) and b) show shell images lit respectively from the upside and downside, while c) surface derivative along y. N.B. in real images the y axis has the opposite direction than y axis in analytical image (vase)



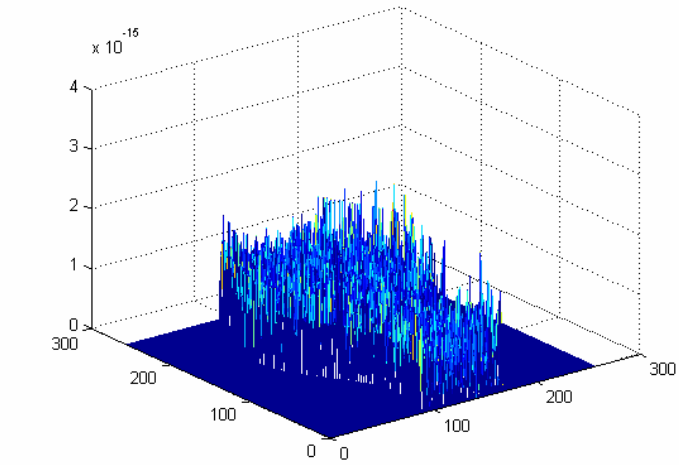
a)



b)



c)



d)

Fig. 3.8 – a) and b) show vase images lit respectively from the upside and downside, while c) surface derivative along y

In this sense the gradient recovery assumes a minor importance than the second step, the gradient integration, which is a fundamental issue to reach the height map starting from the gradient field. However, is the four lights set the best solution and, if yes, in which terms?

Actually the choice of a specific lights set depends on many factors. For instance if we want to decrease acquisition time we could acquire only three color images, which are the minimum data set that permits to compute the gradient field and the albedo. However, if the surface is away from the smoothness condition, more than four images are probably needed to avoid shadows distortions in gradient and albedo. So, it's obvious that the more images we have, the more accurate will be the 3D recovered map, but the more is also the acquisition time. In any case, we need to find a trade-off between these issues. For example, in real-time applications we are bound by hardware acquisition rate and we should choose the best solution in terms of images number and images resolution. The latter depends on which object we have to deal with. In the next section the basic gradient extraction theory and formulas were explained [12].

4. Basic theory – gradient extraction

In this section we discuss the theory of gradient computation. In order to define a proper mathematical model we consider a Lambertian surface behaviour. Many works demonstrate that it's possible, under some assumptions, to separate this special cue to the others (highlight, shadows an so on) [10]. In such a way many non-ideal surfaces can be transformed to be treated as Lambertian. For this reason the following theory is usable in almost every photometric stereo experiment.

The reflectance function in Lambertian case is the scalar product of two vectors, the lighting one and the surface normal:

$$R(p,q)=\frac{1+p_S p+q_S q}{\sqrt{1+p_S^2+q_S^2}\sqrt{1+p^2+q^2}}=\frac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1)\cdot\frac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1) \qquad (4.1)$$

Now, to simplify notation consider this vectors:

$$\begin{cases} l=\dfrac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1) \\ n=\dfrac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1) \end{cases} \qquad (4.2)$$

If we have $k=1,2,\ldots,N$ light sources we can write:

$$I^k=l^k\cdot n \qquad (4.3)$$

This formula doesn't care about surface chromaticity (albedo).

But where could albedo occur and what does its appearance depend on? While we are sure that it occurs in surface microfacets (which are projected onto pixels in image plane) as its absolute property so-called body colour, its appearance could depend on the light colour. Considering this latter factor we have to rewrite the previous equation in this way:

$$\begin{cases} L = \mu l = \mu \dfrac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1) \\ n = \dfrac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1) \end{cases} \tag{4.4}$$

where $\mu = [\mu_R \quad \mu_G \quad \mu_B]^T$, if we consider the separated RGB signals.

Now, from (4.3) and (4.4) we obtain

$$I^k = L^k \cdot n = \begin{bmatrix} \mu_R \\ \mu_G \\ \mu_B \end{bmatrix} \frac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1) \frac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1)^T = \\ = \frac{1}{\sqrt{1+p_S^2+q_S^2}\sqrt{1+p^2+q^2}} \begin{bmatrix} \mu_R(1+p_S p+q_S q) \\ \mu_G(1+p_S p+q_S q) \\ \mu_B(1+p_S p+q_S q) \end{bmatrix} = \mu R(p,q) \tag{4.5}$$

Let now deal with the body colour. As the light colour is a RGB vector multiplied by the light direction vector, in the same way the body colour is a RGB vector multiplied by surface normal. (4.4) becomes

$$\begin{cases} L = \mu l = \mu \dfrac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1) \\ N = \rho n = \rho \dfrac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1) \end{cases} \tag{4.6}$$

where $\rho = [\rho_R \quad \rho_G \quad \rho_B]^T$. In this case the colour acquired image is

$$I^k = L^k \cdot N = \begin{bmatrix} \mu_R \\ \mu_G \\ \mu_B \end{bmatrix} \frac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1) \left(\begin{bmatrix} \rho_R \\ \rho_G \\ \rho_B \end{bmatrix} \frac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1) \right)^T = \\ = \frac{1}{\sqrt{1+p_S^2+q_S^2}\sqrt{1+p^2+q^2}} \begin{bmatrix} \rho_R\mu_R(1+p_S p+q_S q) \\ \rho_G\mu_G(1+p_S p+q_S q) \\ \rho_B\mu_B(1+p_S p+q_S q) \end{bmatrix} = \begin{bmatrix} \rho_R\mu_R \\ \rho_G\mu_G \\ \rho_B\mu_B \end{bmatrix} R(p,q) \tag{4.7}$$

Generally, since the concept of colour is relative to a fixed colour, which is considered the reference one, we could consider system normalized regarding the light colour. In such a way the image equation is:

$$I^k = \rho\left(L^k \cdot n\right) \qquad (4.8)$$

Note that, in the ideal case, it doesn't matter using gray-scale or colour images, and it's obvious after considering last equation. However, if there is some noise in acquired image, the analysis of three cues (RGB) could allow us to understand the original colour vector in the RGB cube [10][12], or, at least, to obtain most accurate 3D map.

Considering (4.8) it's obvious that we need at least three images for having a solution. With three images ($k = 1,2,3$) we have this linear system:

$$\rho n = [L]^{-1} I_0 \qquad (4.9)$$

where $[L] = \begin{bmatrix} L^1 & L^2 & L^3 \end{bmatrix}^T$ and $I_0 = \begin{bmatrix} I^1 & I^2 & I^3 \end{bmatrix}^T$. As well known, if $k > 3$, we should apply the Moore-Penrose inverse matrix:

$$\rho n = \left(\left[L\right]^T \left[L\right]\right)^{-1} \left[L\right]^T I_0 \qquad (4.10)$$

That's all regarding the basic theory of gradient extraction. More complex issues will be treated in next sections.

For the moment, we are interested in comparing this kind of gradient computation and the particular method based on four image acquisitions seen in previous section. We analytically analyse equation form using Moore-Penrose matrix and in the same light set condition than before. With the four light sources used in section 3, where we have

$$\begin{cases} \beta_1 = 0 \\ \beta_2 = \dfrac{\pi}{2} \\ \beta_3 = \pi \\ \beta_4 = -\dfrac{\pi}{2} \end{cases} \tag{4.11}$$

the L matrix will be

$$L = \begin{bmatrix} \sin\alpha & 0 & \cos\alpha \\ 0 & \sin\alpha & \cos\alpha \\ -\sin\alpha & 0 & \cos\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{4.12}$$

so,

$$\rho n = \left([L]^T [L] \right)^{-1} [L]^T I_0 = \frac{1}{4} \begin{bmatrix} 2\csc\alpha & 0 & -2\csc\alpha & 0 \\ 0 & 2\csc\alpha & 0 & -2\csc\alpha \\ \sec\alpha & \sec\alpha & \sec\alpha & \sec\alpha \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} \tag{4.13}$$

This leads to

$$\begin{cases} p = -2\cot\alpha \dfrac{I_1 - I_3}{I_1 + I_2 + I_3 + I_4} \\ q = -2\cot\alpha \dfrac{I_2 - I_4}{I_1 + I_2 + I_3 + I_4} \\ \rho = \dfrac{1}{4}\sec\alpha (I_1 + I_2 + I_3 + I_4)\sqrt{1 + p^2 + q^2} \end{cases} \tag{4.14}$$

In the RGB case we could write these equations in such a way

$$\begin{cases} p_R = p_G = p_B = -2\cot\alpha \dfrac{I_{1_{R,G,B}} - I_{3_{R,G,B}}}{I_{1_{R,G,B}} + I_{2_{R,G,B}} + I_{3_{R,G,B}} + I_{4_{R,G,B}}} \\ q_R = q_G = q_B = -2\cot\alpha \dfrac{I_{2_{R,G,B}} - I_{4_{R,G,B}}}{I_{1_{R,G,B}} + I_{2_{R,G,B}} + I_{3_{R,G,B}} + I_{4_{R,G,B}}} \\ \begin{bmatrix} \rho_R \\ \rho_G \\ \rho_B \end{bmatrix} = \frac{1}{4}\sec(\alpha)\sqrt{1+p^2+q^2} \begin{bmatrix} I_{1_R} + I_{2_R} + I_{3_R} + I_{4_R} \\ I_{1_G} + I_{2_G} + I_{3_G} + I_{4_G} \\ I_{1_B} + I_{2_B} + I_{3_B} + I_{4_B} \end{bmatrix} \end{cases} \tag{4.15}$$

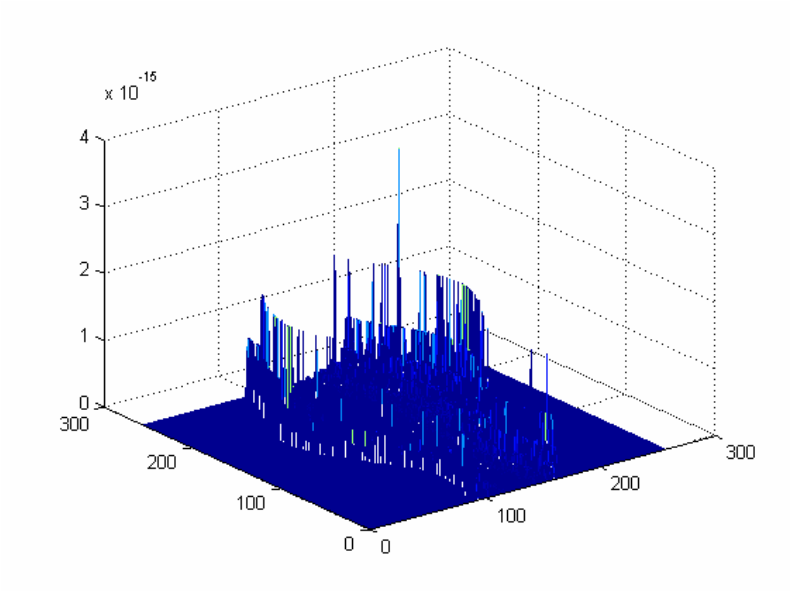
In absence of noise each colour cue produces the same gradient. However, it's very useful, because we could use this behaviour to make the algorithm more robust in presence of noise. Equations (4.14), compared with equations (3.2) and (3.3) tell us that, if there aren't any non-idealities

$$I_1 + I_3 = I_2 + I_4 \tag{4.16}$$

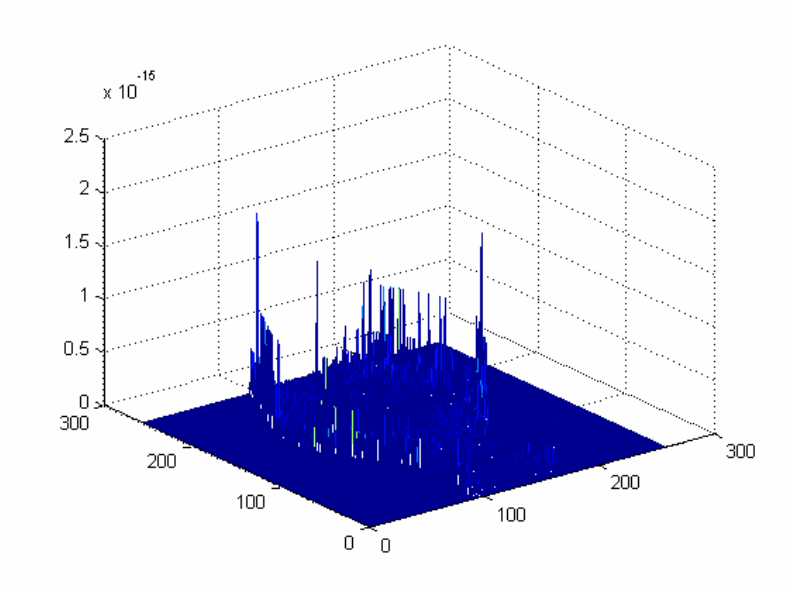
This will be a simple way to detect non-idealities such shadows, highlights or non-parallel rays.

Now, let compare the gradient field of the two used images sets (shell and vase) computed with the latter equations and with (3.2) and (3.3). We should expect the same result with the vase due to the fact that it's an analytical ideal surface, while we should have a very different result in the shell case, most of all in shadows areas.

Fig.4.1 shows what we expected, i.e. the two vase computed gradients are numerically the same. Instead, Fig.4.2 states that the two shell gradient fields are different. Precisely they differ more in the zone in which there are non-idealities. For this reason Fig. 4.3 shows an image where black means less difference; as expected, shadows zone are white.

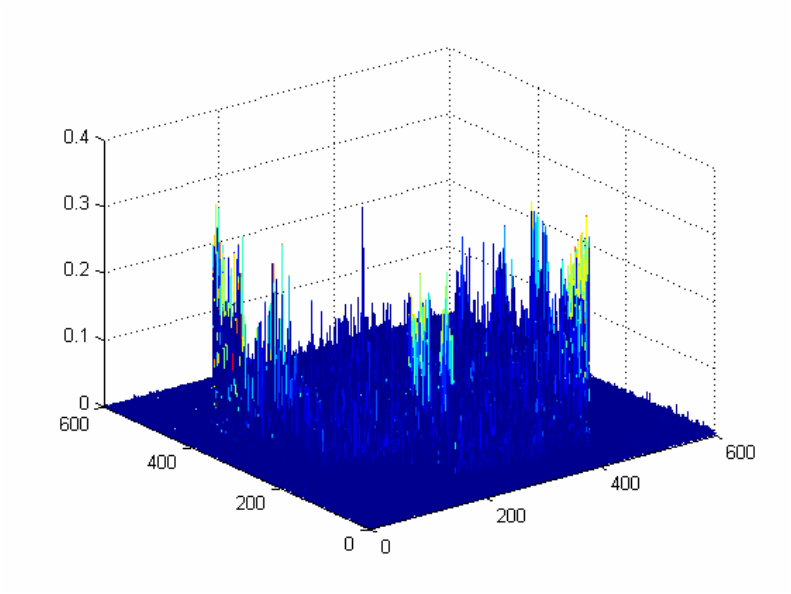


a)

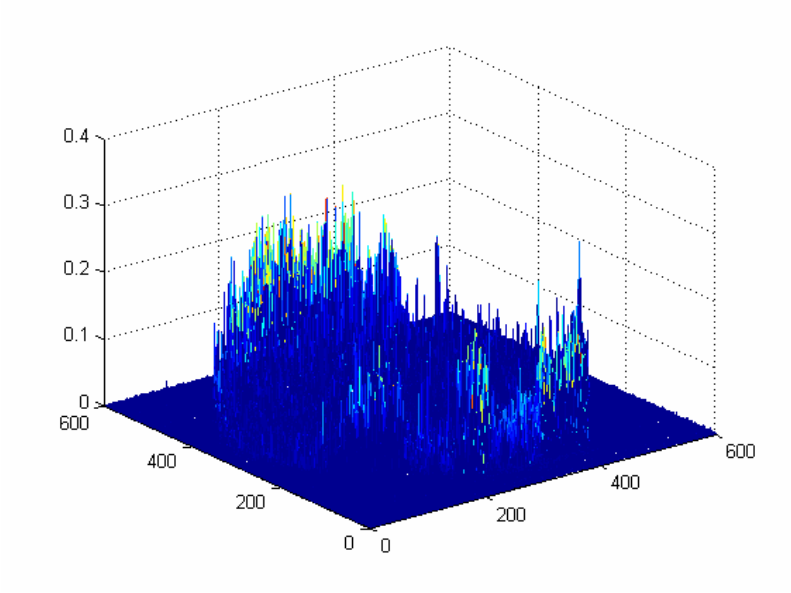


b)

Fig. 4.1 – a) and b) are the differences between the vase x and y derivatives computed with (4.14) and (3.2) with (3.3)

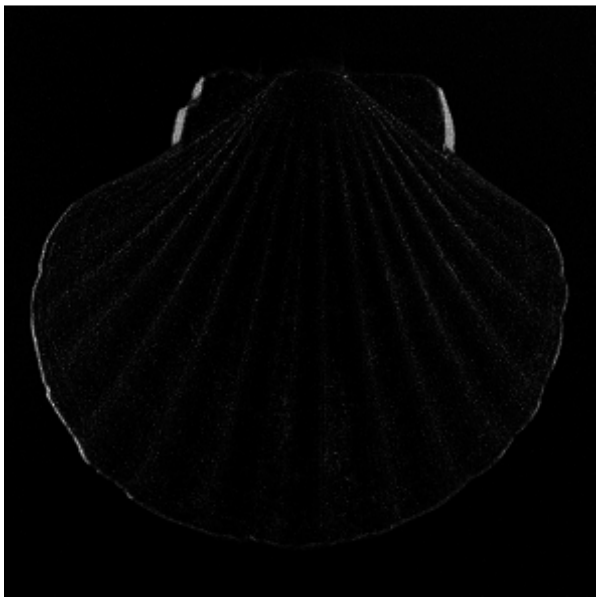


a)

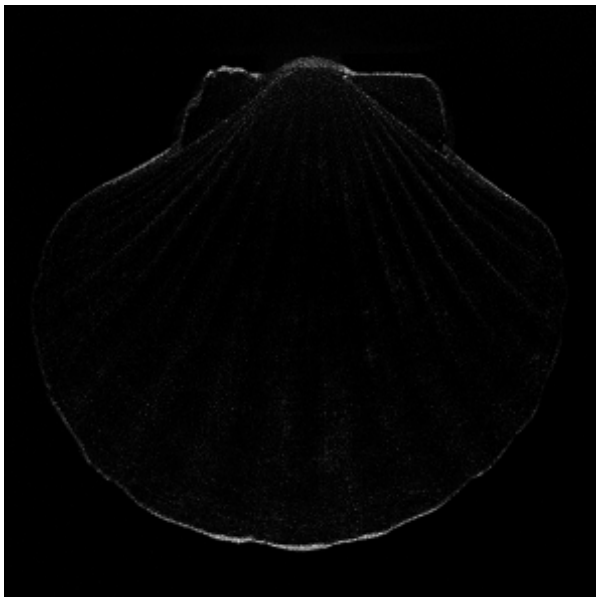


b)

Fig. 4.2 – a) and b) are the differences between the shell x and y derivatives computed with (4.14) and (3.2) with (3.3)



a)



b)

Fig. 4.3 – a) and b) are the difference images between the shell x and y derivatives computed with (4.14) and (3.2) with (3.3)

5. Rays correction

So far we have seen the basic theory of gradient extraction. However we made some assumptions that could not fit the real conditions. If we take care about considering surface non-idealities, there could be also light behaviours that produce gradient and 3D reconstruction biases. In this sense and under particular conditions we have developed a method to correct this structured noise. In this section we study the simple case of point light source near the object and starting from this particular situation we find that it's possible to apply the founded final equations to any case.

We start this analysis assuming that a point light source is lighting a plane, so for all pixels $p = q = 0$. If the light is near the plane we easily note that two surface points with the same (p, q) values will be lit with different alpha. So, the corresponding pixels in plane image have different gray level Fig. 5.1. This is in contrast to (4.8).

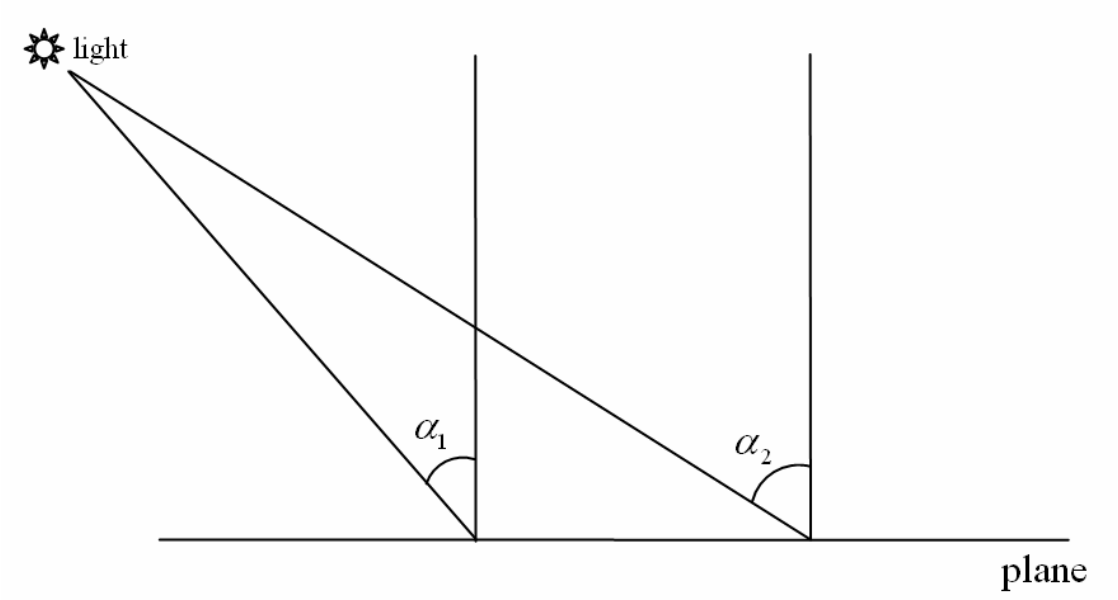


Fig. 5.1 – with a point light source to surface patches with the same gradient are lit with different light directions and then have different pixel gray levels

One thing is clear: for a plane I have a different alpha value for each surface point. We solve this problem (for now in the plane case) using three steps. The first is considering a plane with known albedo (or we could consider its albedo as the reference colour value). Then an average alpha value is found. The last step is to build a correction matrix for the specific light source. After these steps we are able to 3D map and to recover albedo of any plane at the same z

value. It's not really a big result, but, after describing this procedure, we demonstrate that it is possible to improve it under particular but not too binding conditions.

Only in this case we use a different sample instead of shell (it would be vain to consider an ideal surface such the vase), due to the fact that what we are going to explain is more visible with it rather than the shell.

Let consider a white plane, a white light source and gray-scale images (for now we don't care about random noise correction). For this surface, as state above, $p = q = 0$ for all patches, so we can write:

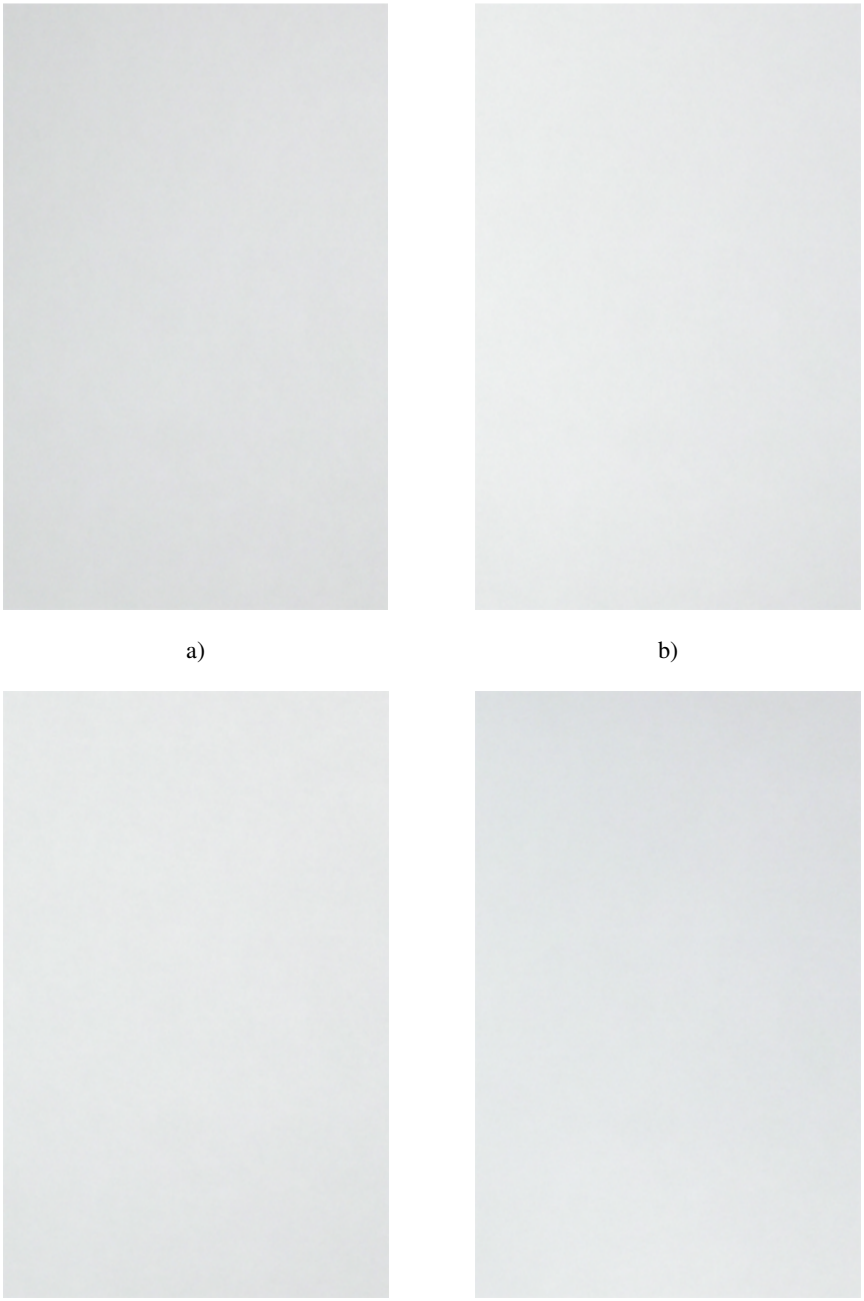
$$I^k = L^k \cdot N = \frac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1)\left(\frac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1)\right)^T\Bigg|_{\substack{p=0\\q=0}} = \tag{5.1}$$

$$= \frac{1}{\sqrt{1+p_S^2+q_S^2}} = \cos\alpha$$

If rays are not parallel we should make a distinction between all pixels so, in matrix terms,

$$\tilde{I}_{i,j}^k = \cos\alpha_{i,j}^k \tag{5.2}$$

If we know the value of alpha we also know the alpha biases in each real image pixel. Now, how we could state the real value of alpha if the light is a point light source? One method should be to consider the alpha between the ray which links light position and the intersection between plane and view direction. However it would be a good method only for point light source and we should choose another approach with a casual light distribution (not only regarding light direction but also regarding light intensity). Instead, knowing the albedo and gradient of the plane, we could measure the average alpha value. Fig. 5.2 shows the same plane lit from four different directions (the same of the previous section). As is easy to see, instead of the ideal case, the gray-level distribution is not uniform in a single image and it's not equal comparing images one another.



a) b)

c) d)

Fig. 5.2 – a),b),c) and d) show the white plane lit respectively from right, left, upside and downside

So, if we consider a NxM matrix the mean alpha value is

$$\alpha_M = \arccos\left(\frac{\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=}^{M} I_{i,j}^{k}}{KNM}\right) \qquad (5.3)$$

These two methods are the approach boundaries. The first is the same as the second with a Dirac function applied in the image center. A reasonable trade-off is adding a weight depending on the distance between the pixel and image center. The equation becomes

$$\alpha_M = \arccos\left(\frac{\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} I_{i,j}^{k}}{K\sum_{i=1}^{N}\sum_{j=2}^{M} w_{i,j}} \right) \qquad (5.4)$$

(N.B. we put K out of the summary because the four weighted matrices are equal each other)

where

$$w_{i,j} = 1 - \sqrt{\frac{\left(\frac{N}{2} - i\right)^2 + \left(\frac{M}{2} - j\right)^2}{\left(\frac{N}{2} - 1\right)^2 + \left(\frac{M}{2} - 1\right)^2}} \qquad (5.5)$$

Once we have computed alpha we build a correction matrix M^k in such a way

$$I^k = M^k \tilde{I}^k \qquad (5.6)$$

so,

$$I_{i,j}^k = M_{i,j}^k \tilde{I}_{i,j}^k = M_{i,j}^k \cos\alpha_{i,j}^k = \cos\alpha_M \qquad (5.7)$$

We have founded a simple equation for the correction matrix

$$M_{i,j}^k = \frac{\cos\alpha_M}{\tilde{I}_{i,j}^k} \qquad (5.8)$$

Fig. 5.3 shows plane images after applying the correction matrices on them. The computed alpha value is $\alpha_M \approx 0.45rad \approx 26°$. Chosen these correction matrices, obviously it doesn't matter if we'll deal with a plane with different albedo, because it is only a gray-level pixel scale factor. In this case we only have a different equation

$$I_{i,j}^k = M_{i,j}^k \tilde{I}_{i,j}^k = M_{i,j}^k \rho \cos\alpha_{i,j}^k = \rho \cos\alpha_M \qquad (5.9)$$

So,

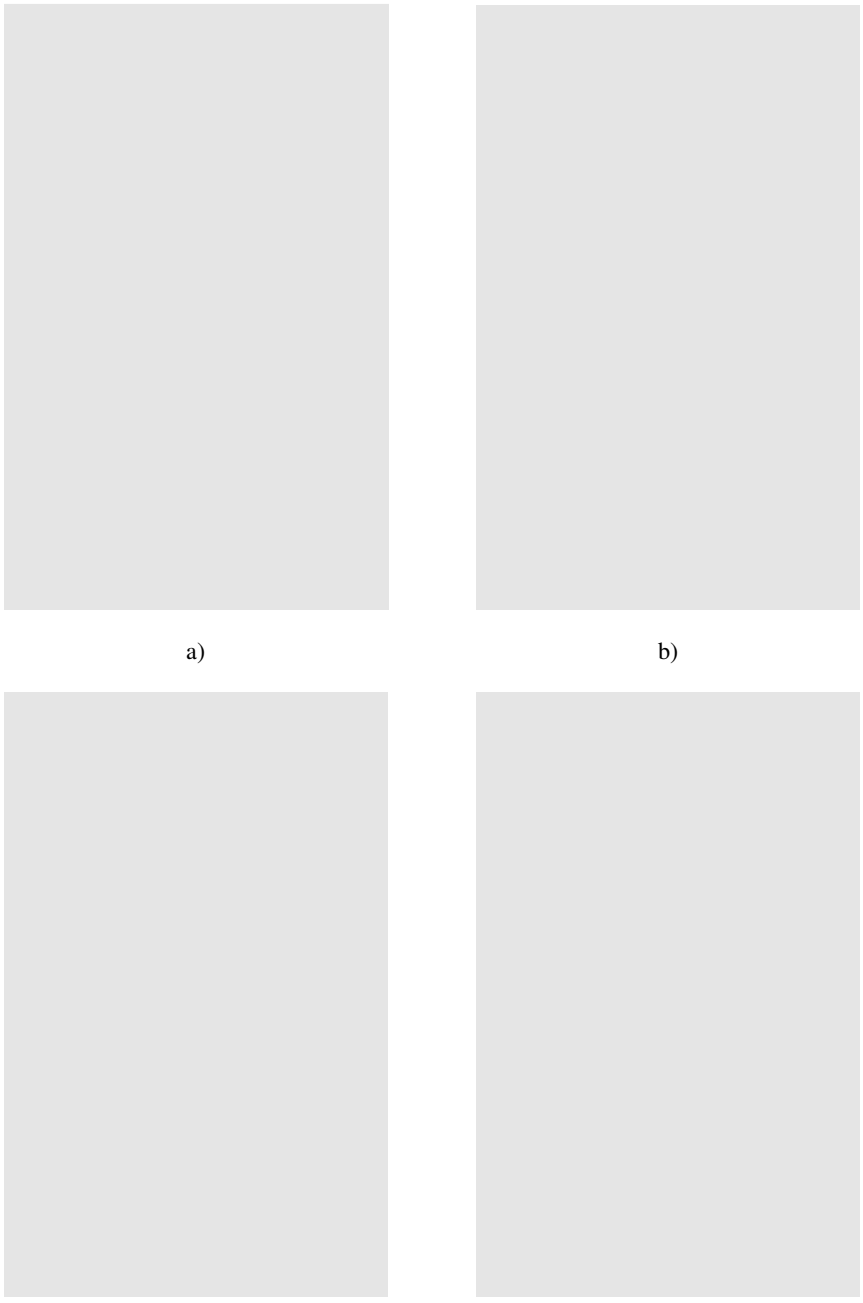
$$\alpha_M = \arccos\left(\frac{\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} \frac{I_{i,j}^k}{\rho}}{K\sum_{i=1}^{N}\sum_{j=2}^{M} w_{i,j}} \right) \qquad (5.10)$$

and

$$M_{i,j}^k = \frac{\rho \cos\alpha_M}{\tilde{I}_{i,j}^k} \qquad (5.11)$$

As we said before this is not a really great result, since if I move the plane forwards or backwards along z axis the alpha correction at the same pixel changes. For instance, if I put an object on the plane and I want to recover its gradient I should consider a different alpha distortion than the plane case. How can I go on to make this result be worth effort?

In Fig. 5.1 we have seen that two pixels involve different light directions, if the light is not far enough to make us consider parallel rays. But, if the light is not too near the plane we could demonstrate that for a little difference in z axis the same correction matrix is valid yet. The only thing we have to study is this z range, in which we could generalize the particular plane correction matrices and use them for any object. Let start to see the geometric problem (Fig. 5.4). I have to find the ∂z that permits me to consider alpha variation numerically zero in the case of point light source. Independently of the surface slope the alpha variation involve a variation in l term. We would want that each element of light vector doesn't change the value out of a specific range.



a) b)

c) d)

Fig. 5.3 – a),b),c) and d) show the white plane lit respectively from right, left, upside and downside after applying the correction matrix on it

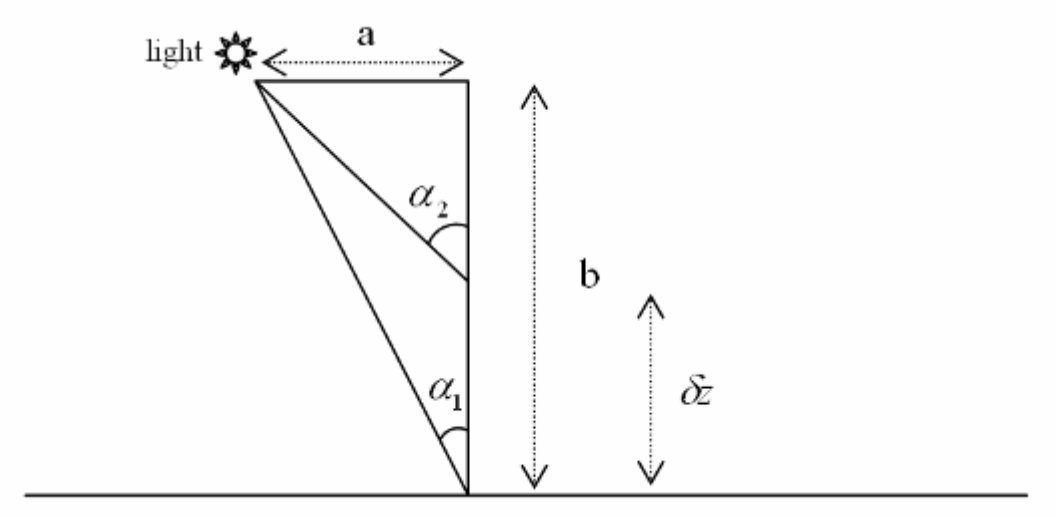


Fig. 5.4 – For applying correction matrices to any object being 3D mapped I should find the variation of z that permits to consider numerically zero the variation of alpha

The involved vector is

$$l = \begin{bmatrix} \sin\alpha\cos\beta & \sin\alpha\sin\beta & \cos\alpha \end{bmatrix} \qquad (5.12)$$

We want that

$$\max\left|\frac{l_1 - l_2}{l_1}\right| \le 1\% \qquad (5.13)$$

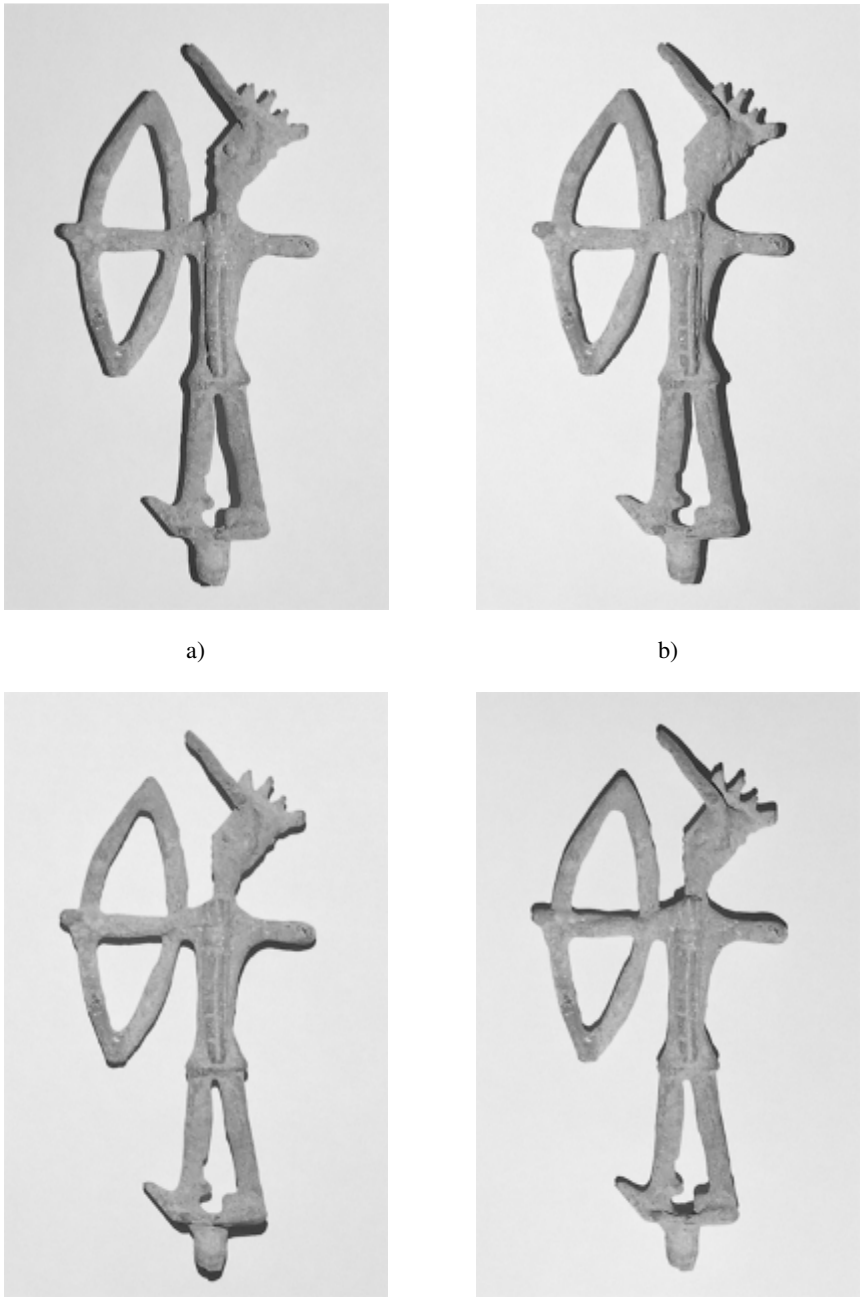
This condition leads to

$$\begin{cases} \dfrac{\sin\alpha_2 - \sin\alpha_1}{\sin\alpha_1} \le 0.01 = \varepsilon \\ \dfrac{\cos\alpha_1 - \cos\alpha_2}{\cos\alpha_1} \le 0.01 = \varepsilon \end{cases} \qquad (5.14)$$

Using Fig. 5.4 we can write

$$\delta z = \frac{2\varepsilon b}{1+\varepsilon} \qquad (5.15)$$

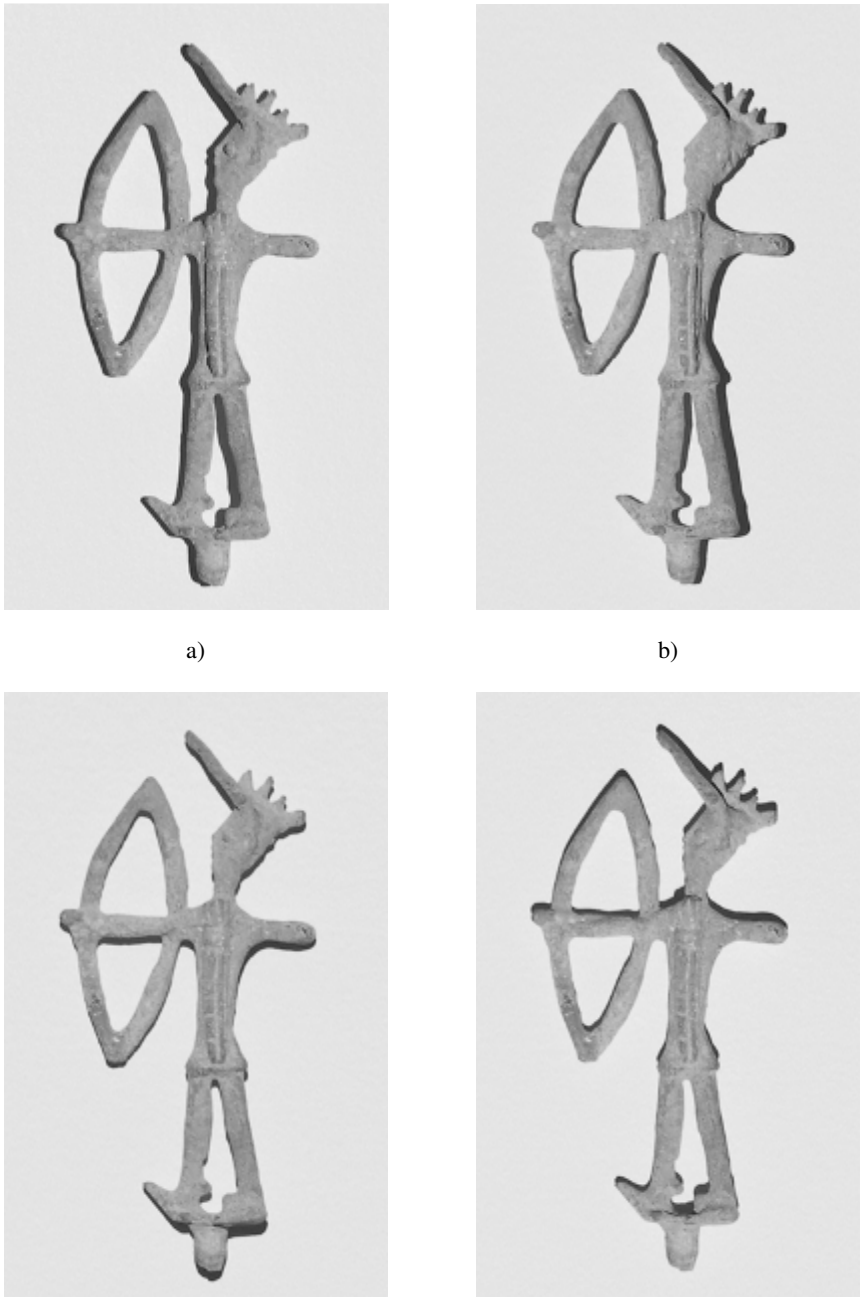
This result means that, if the plane, in which the light sources lie, is one meter and a half far from the plane where I could put any sample on, I could reconstruct a gradient field of a surface that has a z map in almost 3 cm range. This is a very similar condition, comparing with our experiments, so we could apply correction matrices even if there is an object on the original white plane. Fig. 5.5 shows four images of an archaeological sample. Now we apply on them the correction matrices (Fig. 5.6) and we show the results (Fig. 5.7). Even if we didn't mention the integration algorithm yet, in order to understand the effect of this correction approach, we also show the 3D reconstructions using the original images and the corrected ones (Fig. 5.8). As is easy to see, the 3D map using original images is biased: the plane should be a flat surface instead of semi-spherical. On the other hand, the surface computed after using correction matrices appear no biased and the plane under the sample is flat, as it should be.



a) b)

c) d)

Fig. 5.5 – a),b),c) and d) show the archaeological sample lit respectively from right, left, upside and downside before applying the correction matrix on it



a) b)

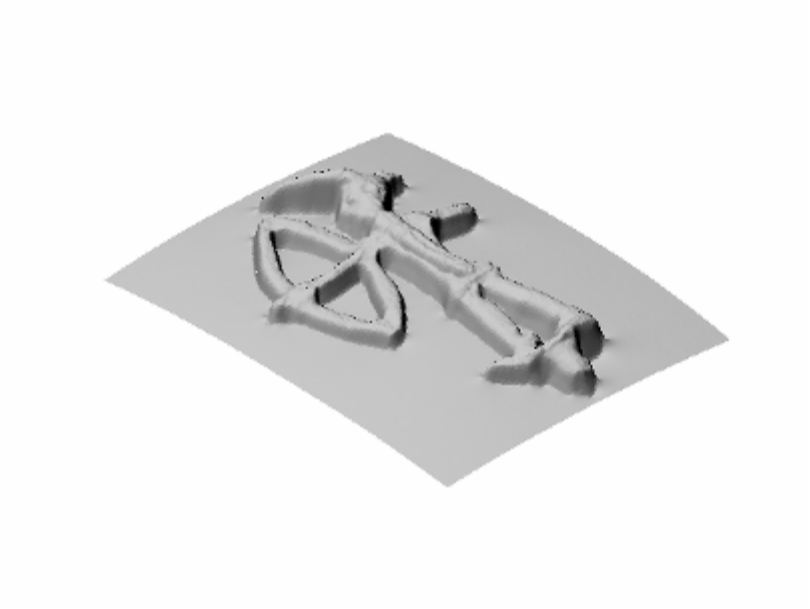
c) d)

Fig. 5.6 – a),b),c) and d) show the archaeological sample lit respectively from right, left, upside and downside after applying the correction matrix on it

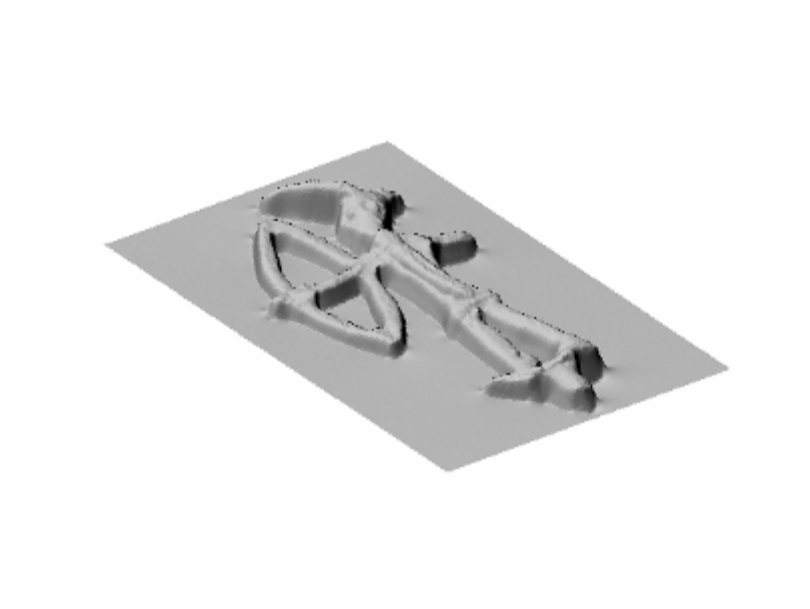
a)

b)

Fig. 5.7 – a), b) show the archaeological sample gradient computed after applying the correction matrix on it



a)



b)

Fig. 5.8 – the archaeological sample 3D map computed a) without and b) after applying the correction matrix on it

6. Gradient Integration

In previous section we have analysed the gradient computation process. Now we start to deal with the second main step, i.e. the integration step, which permits to perform a transformation between the gradient field and the corresponding heights map. In the past, the first test consists in applying the Riemann one dimensional integration, since we had only two photos and one derivative (x or y). The results are not really good, most of all in the real case. Indeed the Riemann approach is not too bad in the ideal case, but it's very noise sensitive, so it becomes unusable with real surfaces. Concerning the vase, for example, the 3D map is quite good in the case of x derivative while is really bad with the y derivative (Fig. 6.1). This depends on the fact that Riemann integration performs only one integration per row (or column) and doesn't make any correlation between columns (or rows). Furthermore, if it doesn't have the boundary condition as initial cue, the first row (or column) starts from zero value. It's not worth developing a Riemann based algorithm that cares about the boundaries. For this reason an ideal surface with the first column equal to zero (as the vase) has a good reconstruction with the Riemann integral applied on the x derivative (Fig. 6.1). This represents the first trial and, for obvious reasons, we immediately abandon this kind of approach. We have read a lot of works and tried some known algorithms concerning this issue and there is a lot of different ways to recover z map from the gradient field, for instance using Fourier transforms [13] or Green's formulas, but so far we believe that the best trade-off between the reconstruction accuracy and computational time is the biconjugate gradient method (BGM) approach [14]. In this section we're going to discuss a particular algorithm we have developed based on BGM, taking care of both surface and boundary integrals. First we'll talk about the matrix problem involved in two-dimensional integration process (Poisson matrix), then we'll show the theory concerning BGM and finally we'll illustrate our modified algorithm.

6.1 Poisson matrix

Poisson equation is the prototypical elliptic equation

$$\frac{\partial^2}{\partial x^2} z + \frac{\partial^2}{\partial y^2} z = \rho(x, y) \qquad (6.1)$$

where $\rho(x, y)$ is the known term. Using the finite-difference and matrix representation it becomes

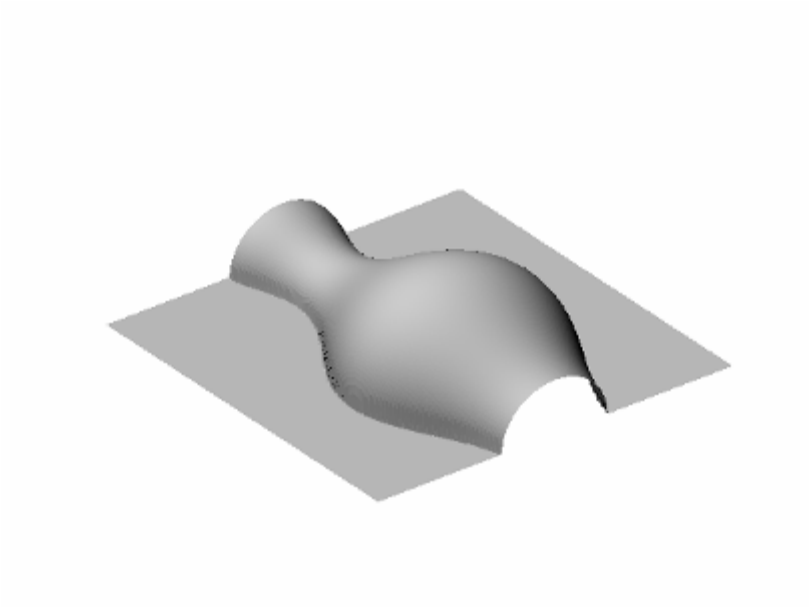
$$\frac{z_{i,j+1} - 2z_{i,j} + z_{i,j-1}}{\Delta^2} + \frac{z_{i+1,j} - 2z_{i,j} + z_{i-1,j}}{\Delta^2} = \rho_{i,j} \qquad (6.2)$$

where Δ is the distance in (x, y) plane between two surface patch centers along x (or y), supposed they are equal. Now let us consider a surface that consists in a NxM matrix. We could write

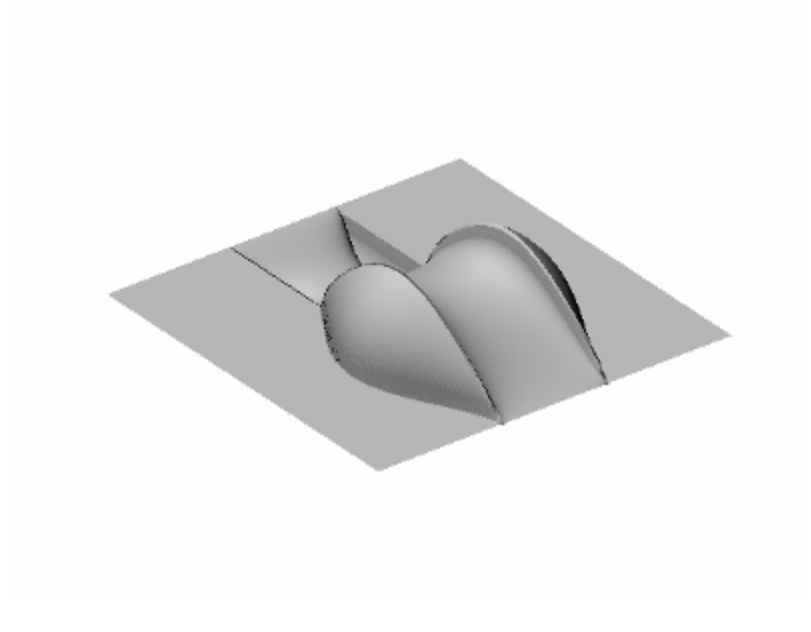
$$z_{i,j-1} + z_{i-1,j} - 4z_{i,j} + z_{i,j+1} + z_{i+1,j} = \Delta^2 \rho_{i,j} \qquad (6.3)$$

for $i = 2 \ldots N-1$ and $j = 2 \ldots M-1$. This leads to a matrix formulation, which involves the so-called "tridiagonal with fringes" matrix. For example, if $(N, M) = (6,6)$, we'll have a 16x16 Poisson matrix.

$$\begin{bmatrix}
-4 & 1 & 0 & 0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
1 & -4 & 1 & \ddots & \ddots & 1 & \ddots & & & & & & & & & \vdots \\
0 & 1 & -4 & 1 & \ddots & \ddots & 1 & \ddots & & & & & & & & \vdots \\
0 & \ddots & 1 & -4 & 0 & \ddots & \ddots & 1 & \ddots & & & & & & & \vdots \\
1 & \ddots & \ddots & 0 & -4 & 1 & \ddots & \ddots & 1 & \ddots & & & & & & \vdots \\
0 & 1 & \ddots & \ddots & 1 & -4 & 1 & \ddots & \ddots & 1 & \ddots & & & & & \vdots \\
\vdots & \ddots & 1 & \ddots & \ddots & 1 & -4 & 1 & \ddots & \ddots & 1 & \ddots & & & & \vdots \\
\vdots & & \ddots & 1 & \ddots & \ddots & 1 & -4 & 0 & \ddots & \ddots & 1 & \ddots & & & \vdots \\
\vdots & & & \ddots & 1 & \ddots & \ddots & 0 & -4 & 1 & \ddots & \ddots & 1 & \ddots & & \vdots \\
\vdots & & & & \ddots & 1 & \ddots & \ddots & 1 & -4 & 1 & \ddots & \ddots & 1 & \ddots & \vdots \\
\vdots & & & & & \ddots & 1 & \ddots & \ddots & 1 & -4 & 1 & \ddots & \ddots & 1 & 0 \\
\vdots & & & & & & \ddots & 1 & \ddots & \ddots & 1 & -4 & 0 & \ddots & \ddots & 1 \\
\vdots & & & & & & & \ddots & 1 & \ddots & \ddots & 0 & -4 & 1 & \ddots & 0 \\
\vdots & & & & & & & & \ddots & 1 & \ddots & \ddots & 1 & -4 & 1 & 0 \\
\vdots & & & & & & & & & \ddots & 1 & \ddots & \ddots & 1 & -4 & 1 \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 0 & 0 & 1 & -4
\end{bmatrix}_{16x16} \qquad (6.4)$$



a)



b)

Fig. 6.1 – 3D map computed applying Riemann integral to a) x and b) y derivative

The unknown term is

$$\begin{bmatrix} z_{2,2} & z_{2,3} & z_{2,4} & z_{2,5} & z_{3,2} & z_{3,3} & z_{3,4} & z_{3,5} & z_{4,2} & z_{4,3} & z_{4,4} & z_{4,5} & z_{5,2} & z_{5,3} & z_{5,4} & z_{5,5} \end{bmatrix}^T \quad (6.5)$$

Instead, the given term is build by $\rho_{i,j}$ and the boundary conditions in such a way

$$\begin{aligned}&\left[\rho_{2,2}\quad \rho_{2,3}\quad \rho_{2,4}\quad \rho_{2,5}\quad \rho_{3,2}\quad \rho_{3,3}\quad \rho_{3,4}\quad \rho_{3,5}\quad \rho_{4,2}\quad \rho_{4,3}\quad \rho_{4,4}\quad \rho_{4,5}\quad \rho_{5,2}\quad \rho_{5,3}\quad \rho_{5,4}\quad \rho_{5,5}\right]^T + \\ &-\left[z_{2,1}\quad 0\quad 0\quad z_{2,6}\quad z_{3,1}\quad 0\quad 0\quad z_{3,6}\quad z_{4,1}\quad 0\quad 0\quad z_{4,6}\quad z_{5,1}\quad 0\quad 0\quad z_{5,6}\right]^T + \\ &-\left[z_{1,2}\quad z_{1,3}\quad z_{1,4}\quad z_{1,5}\quad 0\quad 0\quad 0\quad 0\quad 0\quad 0\quad 0\quad 0\quad z_{6,2}\quad z_{6,3}\quad z_{6,4}\quad z_{6,5}\right]^T\end{aligned} \tag{6.6}$$

Note that the four pixels at the matrix edge are not involved in the surface integration. So, we could generalize these matrices using a generic NxM matrix and using this representation

$$Az = b \tag{6.7}$$

where

$$A_{i,j} \equiv \forall i : i = 2 \ldots N-2, \forall j : j = 2 \ldots M-2 : \begin{cases} \forall i, j : i = j \Leftrightarrow A_{i,j} = -4 \\ \forall i, j, n = 1,2,\ldots,N-3 : i = j-1 \wedge i \neq n(N-2) \Rightarrow A_{i,j} = 1 \\ \forall i, j, n = 1,2,\ldots,N-3 : i = j+1 \wedge i \neq n(N-2)+1 \Rightarrow A_{i,j} = 1 \\ \forall i, j : i = j-M \Rightarrow A_{i,j} = 1 \\ \forall i, j : i = j+M \Rightarrow A_{i,j} = 1 \end{cases} \tag{6.8}$$

and

$$b = \rho - z_{bc} - z_{br} \tag{6.9}$$

where

$$z_{bc\,k,1} \equiv \begin{cases} \forall k, n = 0,1,\ldots N-3 : k = n(M-2)+1 \Rightarrow z_{bc\,k,1} = z_{n+2,1} \\ \forall k, n = 1,\ldots N-2 : k = n(M-2) \Rightarrow z_{bc\,k,1} = z_{n+1,M} \\ \forall k, n_1 = 0,1,\ldots N-3, n_2 = 1,\ldots N-2 : k \neq n_1(M-2)+1 \wedge k \neq n_2(M-2) \Rightarrow z_{bc\,k,1} = 0 \end{cases} \tag{6.10}$$

$$z_{br\,k,1} \equiv \begin{cases} \forall k : k = 1,\ldots M-2 \Rightarrow z_{br\,k,1} = z_{1,k+1} \\ \forall k : k = (M-2)(N-3)+1 \ldots (M-2)(N-2) \Rightarrow z_{br\,k,1} = z_{N,k-(M-2)(N-3)+1} \\ \forall k : k \neq 1,\ldots M-2 \wedge k \neq (M-2)(N-3)+1 \ldots (M-2)(N-2) \Rightarrow z_{br\,k,1} = 0 \end{cases} \tag{6.11}$$

For solving (6.7) equation it's possible to use a lot of numerical methods, but we should narrow down the possible algorithm taking care that most of all involve A inverse matrix computation. Since A matrix has $((N-2)(M-2),(N-2)(M-2))$ dimension, even with small value of N and M, the calculation of the inverse could be computationally very hard. The BGM avoids the inverse computation but involves the original sparse matrix storage. For the moment we shall explain this algorithm and then we'll modify it to avoid also its storage in hardware memory, increasing computational time.

6.2 BGM

The conjugate gradient algorithm is based on the minimization of this formula [14]:

$$f(z)=\frac{1}{2}z\cdot A\cdot z-b\cdot z \qquad (6.12)$$

It's minimized when

$$\nabla f = A\cdot z-b=0 \qquad (6.13)$$

So, it's is minimized for the z values that solves (6.7). The generic numerical algorithm (known as biconjugate gradient method) starts supplying initial vectors r_1 and $\bar{r}_1$ (which have the same z dimension), the first guess z_0 and set $p_1 = r_1$ and $\bar{p}_1 = \bar{r}_1$.

Then, it carries out this recurrence

$$\begin{aligned}
\alpha_k &= \frac{\bar{r}_k\cdot r_k}{\bar{p}_k\cdot A\cdot p_k}\\
r_{k+1} &= r_k-\alpha_k A\cdot p_k\\
\bar{r}_{k+1} &= \bar{r}_k-\alpha_k A^T\cdot \bar{p}_k\\
\beta_k &= \frac{\bar{r}_{k+1}\cdot r_{k+1}}{\bar{r}_k\cdot r_k}\\
p_{k+1} &= r_{k+1}+\beta_k p_k\\
\bar{p}_{k+1} &= \bar{r}_{k+1}+\beta_k \bar{p}_k\\
z_{k+1} &= z_k+\alpha_k p_k
\end{aligned} \qquad (6.14)$$

In our case A is symmetric so we can set $\bar{r}_k = r_k$ and $\bar{p}_k = p_k$. Furthermore we choose $z_0 = 0$, which means that the residual $r_0 = b$. The recurrence becomes

$$\begin{aligned}
\alpha_k &= \frac{r_k \cdot r_k}{p_k \cdot A \cdot p_k} \\
r_{k+1} &= r_k - \alpha_k A \cdot p_k \\
\beta_k &= \frac{r_{k+1} \cdot r_{k+1}}{r_k \cdot r_k} \\
p_{k+1} &= r_{k+1} + \beta_k p_k \\
z_{k+1} &= z_k + \alpha_k p_k
\end{aligned} \qquad (6.15)$$

6.3 Our algorithm

So far we have seen the two-dimensional integration problem and the numerical method to solve it. However, in real photometric stereo case, we haven't any given boundary condition and we really don't want to store a big matrix, even if it's sparse. In fact, if we have a common 512x512 image the A matrix will be 262144x262144 matrix with more than 1 million non-zero elements. We improve this algorithm performing two tasks, first we use the conjugate method to execute a boundary integration and then we avoid storing the sparse matrix with a simple algorithm that compute $p_k \cdot A \cdot p_k$.

6.3.1 Boundary integration

Supposed the derivatives along x and y are two NxN matrices. A particular case of Poisson's formula is used to compute the depth map in the matrix contour. First of all the boundary is considered as a linear vector

$$z_C = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,N-1} & z_{1,N} & z_{2,N} & \cdots & z_{N-1,N} & z_{N,N} & z_{N,N-1} & \cdots & z_{N,2} & z_{N,1} & z_{N-1,1} & \cdots & z_{2,1} & z_{1,1} \end{bmatrix}^T \qquad (6.16)$$

Then the equation (6.2) is rewritten considering only one derivative along the direction of the contour vector

$$\left.\frac{\partial^2 z}{\partial u^2}\right|_i = \frac{z_{C\,i+1} - 2z_{C\,i} + z_{C\,i-1}}{\Delta^2} = \rho_i \qquad (6.17)$$

where ρ_i depends on the surface derivatives along x and y and its value is

$$\rho_i = \begin{cases} p_{1,K} - p_{1,K+1}, K = 2 \ldots N-1, i = 1 \ldots N-2 \\ q_{1,N} - p_{1,N-1}, i = N-1 \\ q_{K,N} - q_{K-1,N}, K = 2 \ldots N-1, i = N \ldots 2N-3 \\ -p_{N,N-1} - q_{N-1,N}, i = 2N-2 \\ -p_{N,K-1} + p_{N,K}, K = N-1 \ldots 2, i = 2N-1 \ldots 3N-4 \\ -q_{N-1,1} + p_{N,1}, i = 3N-3 \\ -q_{K-1,1} + q_{K,1}, K = N-1 \ldots 2, i = 3N-2 \ldots 4N-5 \end{cases} \quad (6.18)$$

Note that the first and the last elements of z_C vector are the same and, since an integration involves an additive constant, we could set $z_{1,1} = 0$. This procedure leads to an A matrix with the diagonal element equal to -2 and, for each row the element value before and after -2 is 1.

6.3.2 Surface integration

Using previous results as BGM algorithm cues, we have to analyse only the known term of (6.7). This is the sum of the second derivatives but, in our experiments, we have the first derivative. Using finite-difference approximation we put

$$\begin{cases} \dfrac{\partial^2 z}{\partial x^2}_{i,j} = \dfrac{\partial p}{\partial x}_{i,j} = p_{i,j+1} - p_{i,j} \\ \dfrac{\partial^2 z}{\partial y^2}_{i,j} = \dfrac{\partial q}{\partial y}_{i,j} = q_{i,j+1} - q_{i,j} \end{cases} \quad (6.19)$$

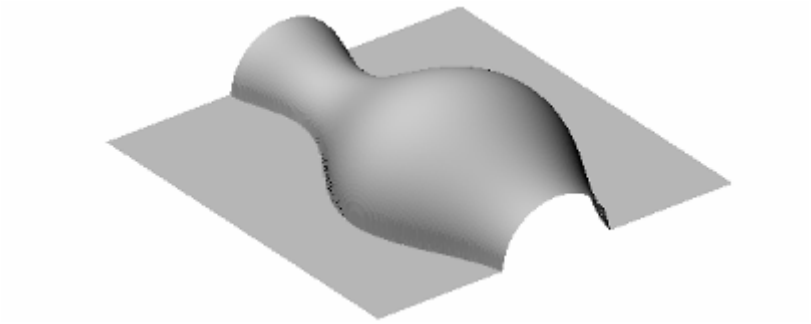
State this, we could write the known term as $b = \rho - z_{bc} - z_{br}$, where z_{bc} and z_{br} are the same as before and ρ is

$$\rho_{i,j} = p_{i,j+1} - p_{i,j} + q_{i,j+1} - q_{i,j} \quad (6.20)$$

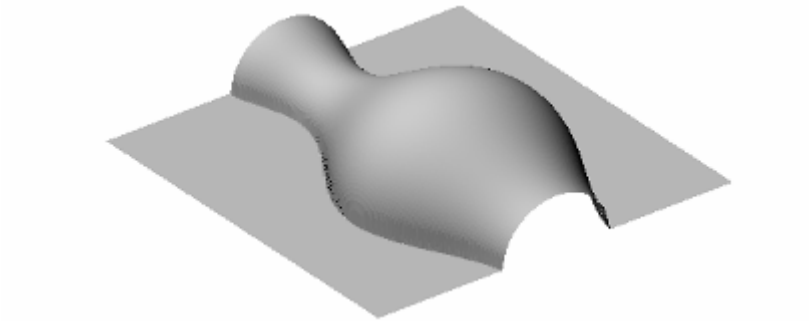
The last thing we have to discuss is the procedure that permits us to avoid A matrix storage. We said that we perfectly know the A matrix and, except little differences, the n+1 row is equal to the shifted n row. If we manage these little differences we could perform the product only with a for loop without storing the sparse matrix. These differences are the zero values that interrupt the diagonal with "1" values adjacent to matrix diagonal. These values occur with a regular recurrence, which depends on the number of columns in the original acquired images. So, every change is predictable and there's no problem to perform this operation. In this way we could reach an algorithm performance better than common algorithm used in professional software for computation.

Now we show the result of this procedure in the analytical and real test cases (vase and shell) (Fig. 6.2 and 6.3). As we explain in section 4, the algorithm recovers the albedo information too. Fig. 6.3 shows the shell reconstructed surface with its applied albedo.

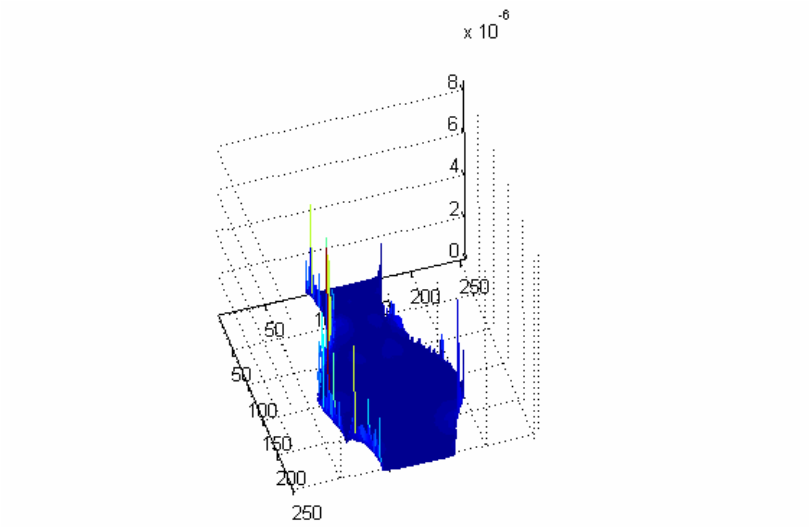
Then, we investigate algorithm changed version to improve its performances and, among many trial, we found that the iteration number (hence the computational time) could be decreased using a 9 pixels kernel instead of Poisson's 5p kernel. Since the derivatives along all direction could be written as a linear combination of x and y derivatives, if we consider also the derivatives along the diagonals of a surface matrix, we'll have a Poisson's equation that include all pixels in a 3x3 kernel centered on (i, j) coordinates. Fig. 6.4 shows a comparison between 5-kernel and 9-kernel algorithm performances in terms of number of iteration and computational time.



a)

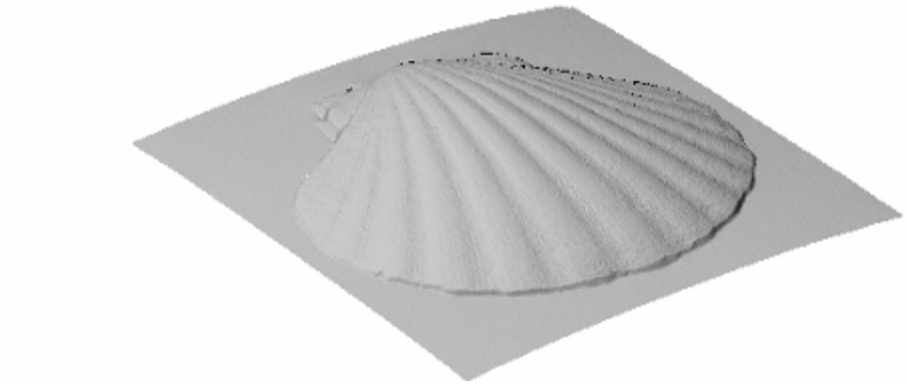


b)



c)

Fig. 6.2 – a) Vase 3D map computed applying BGM integral, b) the original surface and c) the height error %.

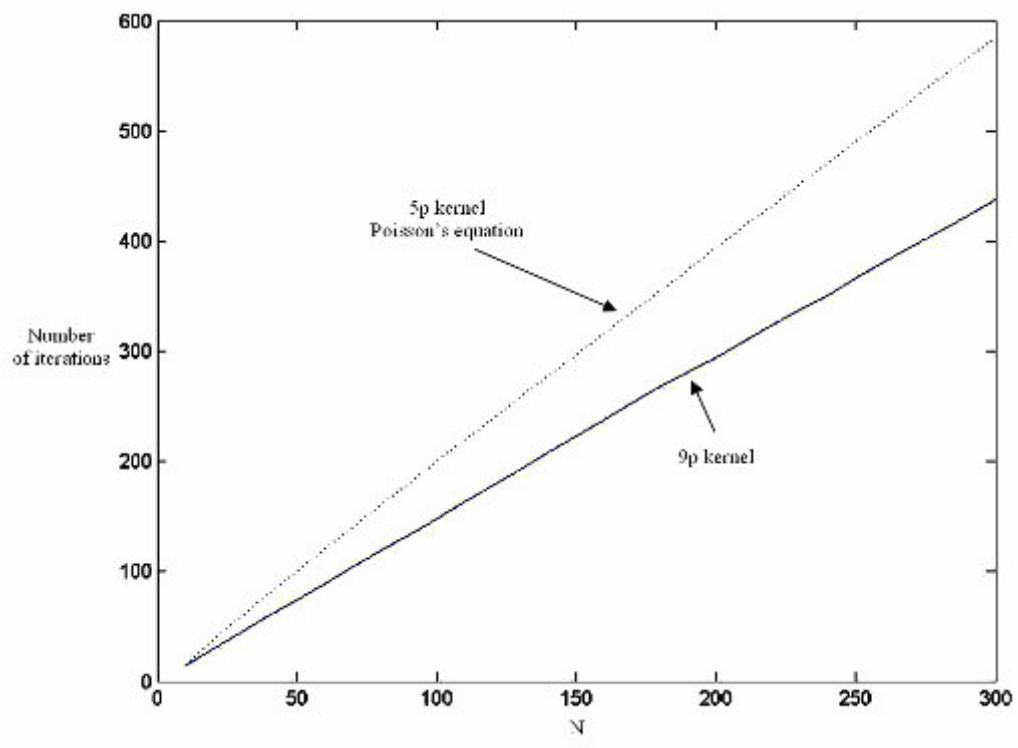


a)

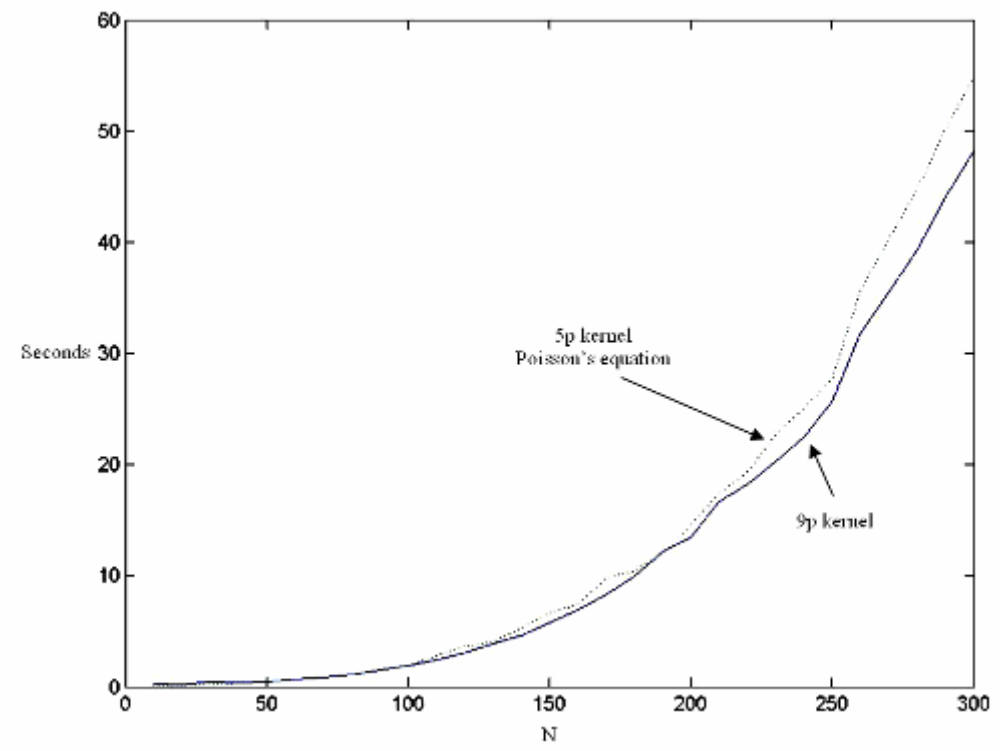
b)

c)

Fig. 6.3 – a), b) shell 3D map computed applying BGM integral, c) the reconstructed surface with applied albedo map.



a)



b)

Fig. 6.4 – a), b) show the performance differences, between our algorithm based on Poisson's equation (5p kernel) and the modified 9p kernel algorithm, respectively regarding the number of iterations and computational time. In abscissas there is the dimension N of a NxN surface matrix.

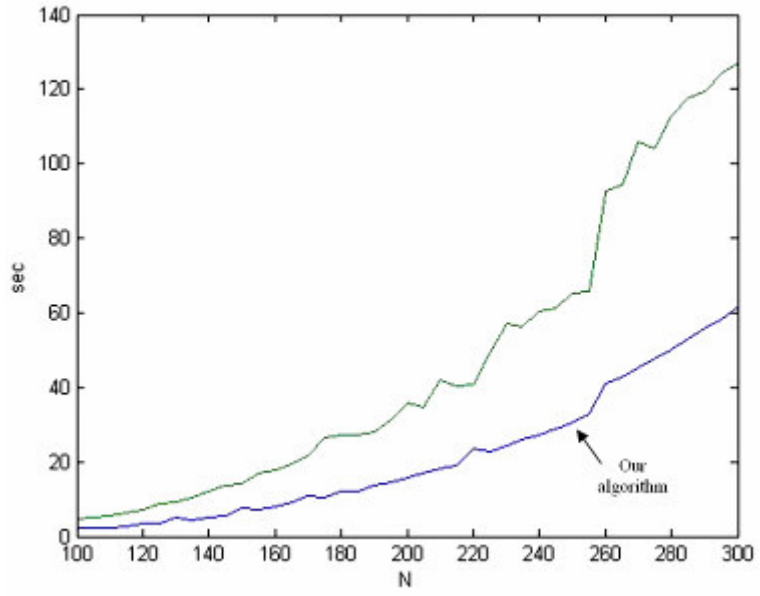
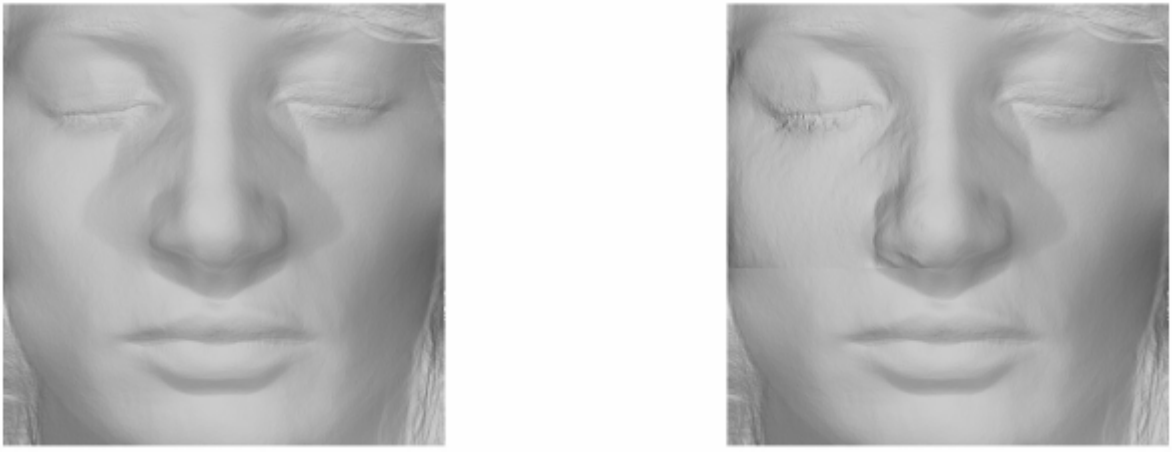


Fig. 6.5 – A comparison between our algorithm and a famous calculus software in solving BG iteration process. In abscissas there is the dimension N of a NxN surface matrix.

These improvements make our algorithm working better than commercial software in solving Poisson's and modified Poisson's algorithms. We find that our algorithm takes less time to compute the biconjugate gradient process than a famous software (that we are not going to mention) (Fig.6.5).

There is another way to improve the computational time regarding the 3D map of a surface. Many papers deal with Photometric stereo issue, but most of all take as initial cue the boundary conditions, which is a really binding thing. We are able to find a 3D map without any information but the gradient, and this is a special feature that allows us to more decrease the time taken to have surface heights. We have only to take not the entire image in a single step, but part of it. After computing the boundary we reconstruct the boundary of a surface region (e.g. the upper left quarter); starting from the known boundary we have to compute only two edges of this subsurface. Then, its surface integral is performed and we could continue finding the 3D map of the other regions. Note that, after this step, if we want to reconstruct the upper right quarter we already know the heights of three of its edge, and so on. The point of fact is that the time needed to work the entire surface out is more than the time taken for calculating four sub-surfaces. Let us consider a NxN matrix with $N = 600$. Our algorithm takes more than seven minutes (about seven minutes and a half) for the entire surface, while it takes about one second for a nxn sub-matrix with $n = 100$. So, with this *"divide et impera"* approach, we spend about 40 seconds instead of 450. If we consider a nxn with $n = 50$, the entire surface will be reconstructed in about 30 seconds. We decrease the computational time by 94%. This approach could be very useful regarding the real-time issue, which could clear the way for new innovative applications.

The possibility of having an algorithm that compute boundary of any region with any topological (two-dimensional) features allows us to solve one of the most important problems: the biases of shadow region. There are some papers [12] that state the capability of identifying shadow zones and robustly computing the real gradient without any distortion. But the shadow zone contains also a lot of information in the case of a non-ideal surface, most of all in the presence of discontinuities, which are the "devil" in photometric stereo. In such cases shadows can help us first to recognize discontinuities and then to reconstruct the right gradient and the right surface. Consider a lit cube lying on a planar surface. It produces a cast shadow on the plane. If we applied one algorithm to remove the shadows effect in gradient field we'll have an equal to zero gradient and the reconstructed surface will be a plane. This is one instance which shows both the lack of many algorithms in managing shadows and the great information contained in them. As an example of applying a simple algorithm to right reconstruct shadow zones without losing the information held in them, let see Fig. 6.6, where we use the capabilities of the curvilinear algorithm (one-dimensional Poisson's approach) to reconstruct a self-shadow in a facial 3D map.



a) b)

Fig. 6.6 – Comparison between facial reconstruction a) without shadow managing and b) removing one of the cast shadows due to the nose

7. Applications

The most wonderful feature in this kind of research is that PS could be applied to a large number of samples and it works not only for near Lambertian surface and using common lights, but also with different kind of images and with surfaces that is non-Lambertian. For instance we reconstruct objects acquiring their scanning electron microscope (SEM) images. Another example is face 3D map; while the skin is approximately Lambertian, the eye is right reconstructed, even if we couldn't consider it Lambertian at all. These are to examples that demonstrate the adaptability of PS to many real cases, even if they are really far from the ideal conditions. This fact proves the developed algorithm is robust regarding most of the distortions causes. Due to this robustness we have applied it to many research and application fields and in the following section we separately analyse each of them:

- archaeology
- biometrics
- graphology
- huge and far objects 3D map
- police inquiry
- SEM samples
- biology
- polynomial texture map (PTM) samples

Furthermore, during these experiments we need a software to manage some data. So we had developed a visualization tool and a software that controls the acquisition of SEM images (section 7.2). We'll broadly show them and their behaviour.

7.1 Optical applications

Archaeology is the first application field we have dealt with during our research. The first sample was a double of a Sumerian tablet, with a great number of incisions on it. We were surprised to note that the 3D map is perfectly readable even if in the first experiments we use a surface matrix with low resolution (Fig. 7.1 and Fig. 7.2). The effect of the heights with albedo separation is evident if we compare the 3D map with and without the chromatic information.

a) b)

c)

Fig. 7.1 – a) one of Sumerian tablet (front) original images, b) 3D map and c) 3D map with albedo

a) b)

c)

Fig. 7.2 – a) one of Sumerian tablet (rear) original images, b) 3D map and c) 3D map with albedo

Concerning archaeological issues a very important application is stratigraphy. Using our algorithm archaeologists can 3D map every step of stratigraphic process, avoiding the needed drawing task. This makes this procedure more accurate and reliable. Fig. 7.3 shows a simulation of a stratigraphic operation supposing we have three surfaces.

a)

b)

c)

Fig. 7.3 – These are three stratigraphic surfaces. a) is the first layer, b) the first and the second layers and c) all layers.

One feature of our algorithm is the separated extraction of heights and chromatic information. This proves to be very useful in archaeology when an inscription is illegible due to its dirtiness. Fig. 7.4 shows a headstone completely impossible to read, while Fig. 7.5 demonstrates that, removing the albedo, with a 3D map we are able to read the original Roman inscriptions.

Fig. 7.4 – A completely illegible headstone

a)

b)

Fig. 7.5 – a) An image of a part of the headstone and b) its 3D map in which we are able to read the original inscriptions

Another important application is biometrics. In this section we are going to show the capabilities of PS in 3D mapping part of human body, useful, for instance, for security application. The first samples we illustrate are facial reconstructions (N.B. we have human guinea-pig leave to public her photo). In Fig. 7.6 we have a face with open eyes. This demonstrates that even non-ideal surfaces could be considered almost Lambertian due to the algorithm robustness regarding non-idealities.

Fig. 7.6 – A face 3D map with right reconstruction of eyes zone

In this sample we could see that the face is reconstructed properly but a lot of distortions appear in the hair. Depending on the specific application (for instance security) we could consider a part of image in which we are able to manage all non-ideal elements and to avoid surface biases. The following figure (Fig. 7.7) proves that it's possible to use our algorithm to 3D map fingerprints. This feature is really important, because generally fingerprints identification is performed with two dimensional data. Possibility to use three-dimensional information improves the identification accuracy and decreases a lot potential errors.

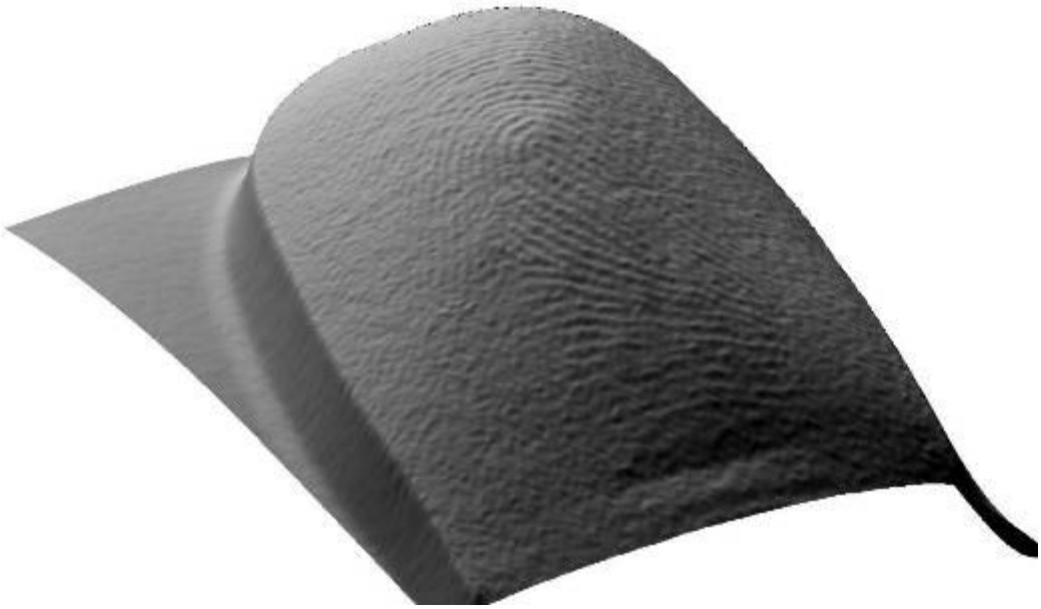


Fig. 7.7 – A fingerprint 3D map

Graphology is the science of human identification by his handwriting. Broadly it uses letter shapes and ink intensity to identify the author of any writing; all of these instruments are two-dimensional. Even in this case having three-dimensional information can help in identifying task. We have tried our algorithm reconstructing two superimposed drawn lines (Fig. 7.8). Using this kind of information we immediately see that one line is deeper than the other, so the man who writes in this paper has used different effort drawing the lines. This could be a good information depending on what somebody is searching for.

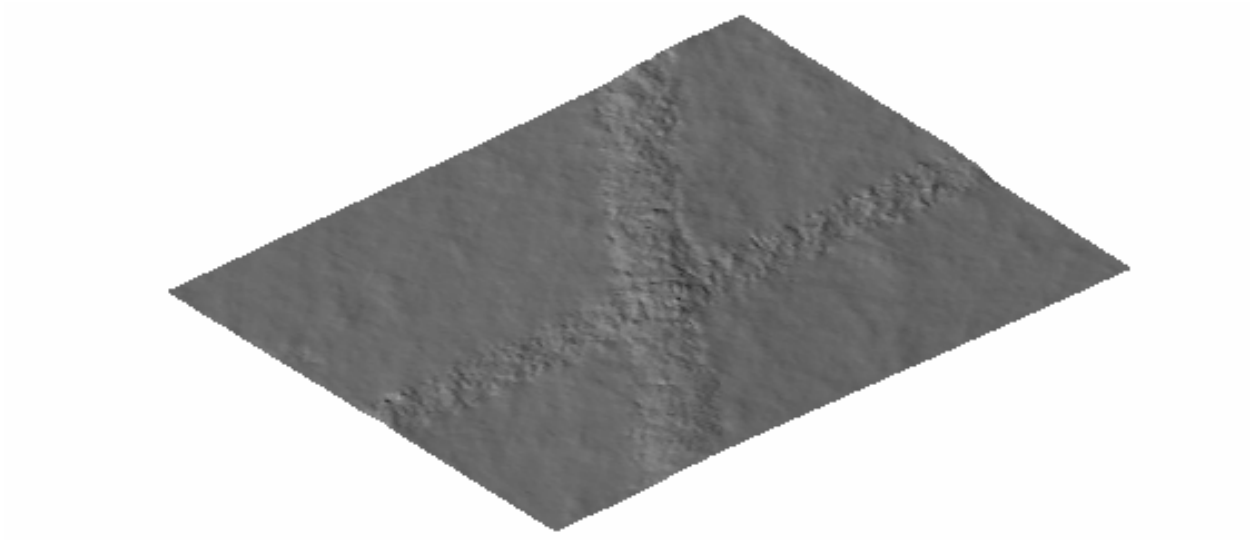
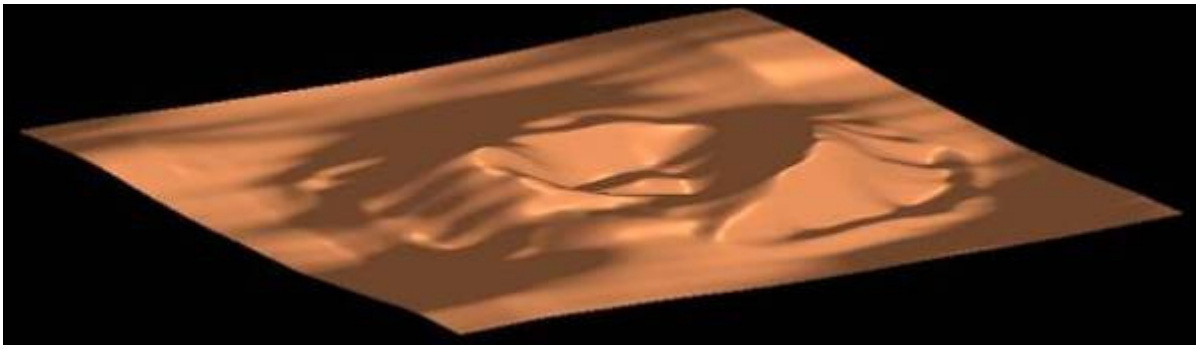


Fig. 7.8 – A 3D map of two superimposed drawn lines

Finally the last kind of application concerning optical domain is the 3D map, using solar light, of object at a great distance from the viewer. In Fig. 7.9 we show the 3D reconstruction of a moon crater.

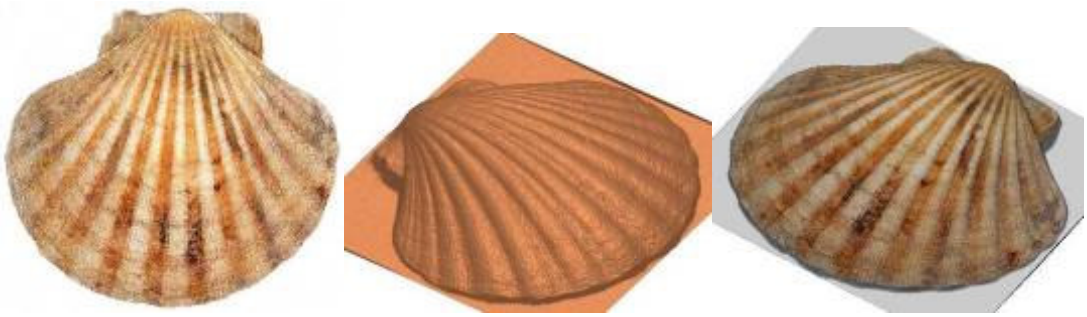
Instead, Fig. 7.10 represents a summarise of the capabilities of our method.

a)



b)

Fig. 7.9 – A moon crater. Original image a) and reconstructed detail b) by 2-source (sunlight at midnight of two successive days) Photometric Stereo.

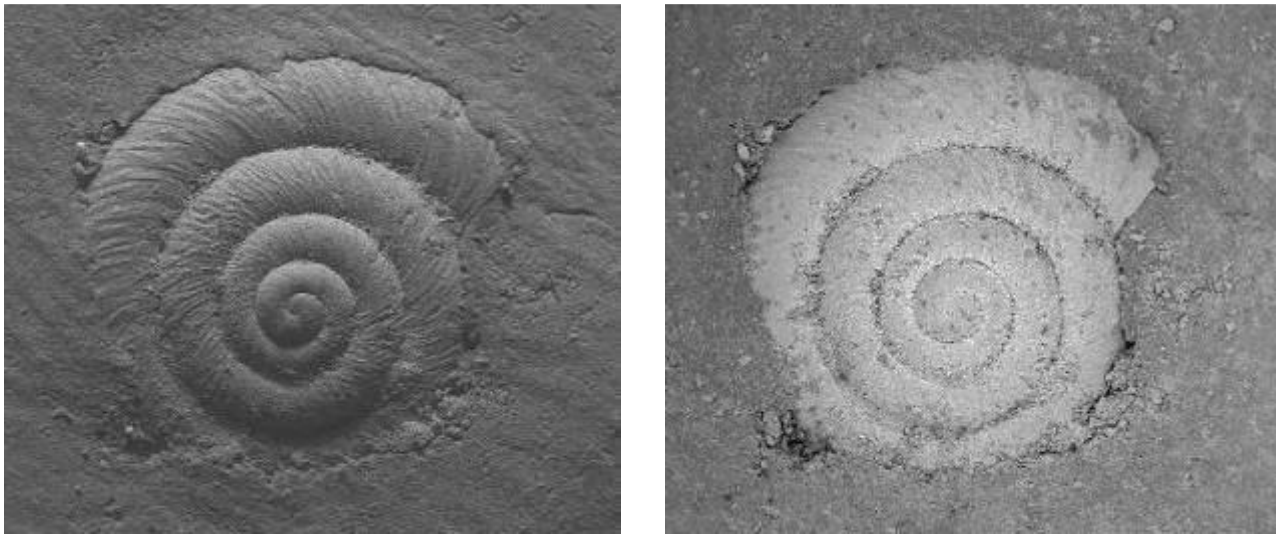


a) b) c)

Fig. 7.10 – Pure albedo a), pure geometry b), and rendered 3D reconstruction c) of a shell by Photometric Stereo. These are the general results after applying our algorithm to a sample in optical domain

7.2 Scanning Electron Microscope

So far we have dealt with optical images and we have considered surfaces behaviour as Lambertian. There is another type of images that, even if it's physically completely different from common images, permits to consider the surface in the same Lambertian way. They are scanning electron microscope (SEM) images. Obviously it increases the possible application field and as we demonstrate that it's possible to reconstruct huge and far object using, for instance, solar light, now we can say we are able to investigate the microscopic world. In the following we'll see many exemplars of typical SEM samples but, before doing this, we should illustrate the procedure to acquired SEM images for 3D reconstruction algorithm. Even if the image acquisition system is very different from many point of view we need the same images set. So, not considering the nature of SEM signal, the unique difference is that SEM images are gray-level matrices and the concept of albedo in this data is not the real chromatic property. In scanning electron microscope albedo variations represent material variations along the surface. Giving a closer look to the PS-based 3D recovery method, it consists of the acquisition of 4 Back-Scattered (BS) images, which allow us to compute surface gradient, and, since we have to rotate the sample under observation due to the fact that we have only one light source (in SEM the corresponding light source is the electron detector), and since the rotation is not perfectly performed by the micro mechanical system in SEM, we need to align acquired images to give our algorithm them. Stated this, to acquire images we need to perform the following procedure, due to the intrinsic microscope behaviour.



a) b)

Fig. 7.11 – a) BS and b) IS images of the same frame scan.

First of all the subject is framed, magnified and focused. Then both brightness and contrast are separately set for the two detectors to be used, i.e. the off-axis BackScattered (BS) electron detector and the complete circular axial

BS detector, which produce isotropic shading (we indicated them as IS images). In Fig. 7.11 you can see the BS and IS images related to the first image acquisition. So, after framing these two images of the object we want to 3D map, we start this sequence

1) Acquisition of a pair BS and IS images, as in fig. 7.11.
2) mechanical rotation of the specimen by 90°
3) Acquisition of a reduced IS image in the central area, that is first counter-rotated by 90°, and then

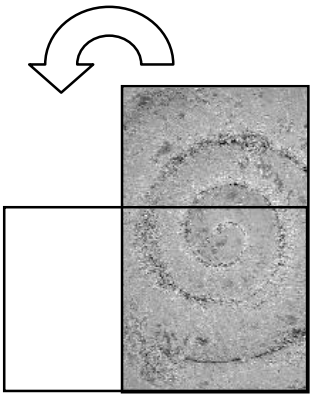


Fig. 7.12 – reduced IS image counter-clockwise rotated by 90°

4) cross-correlate with the previous IS image. This gives the $(\Delta x, \Delta y)$ that we use to re-align the image in SEM acquisition display

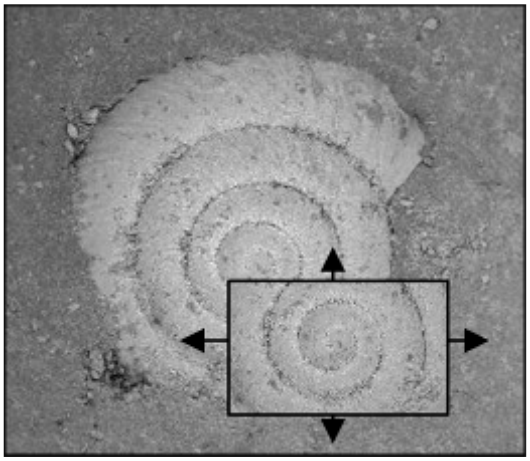


Fig. 7.13 – cross-correlate with the previous IS image

5) The new image center of the rotated specimen is mechanically brought back by $(-\Delta x, -\Delta y)$ as close as possible to the previous centre
6) Steps 1-4 are repeated 3 times

At the end of the process (which takes few minutes, including the image acquisition), a twin set of 4+4 images is stored (Fig. 7.14).

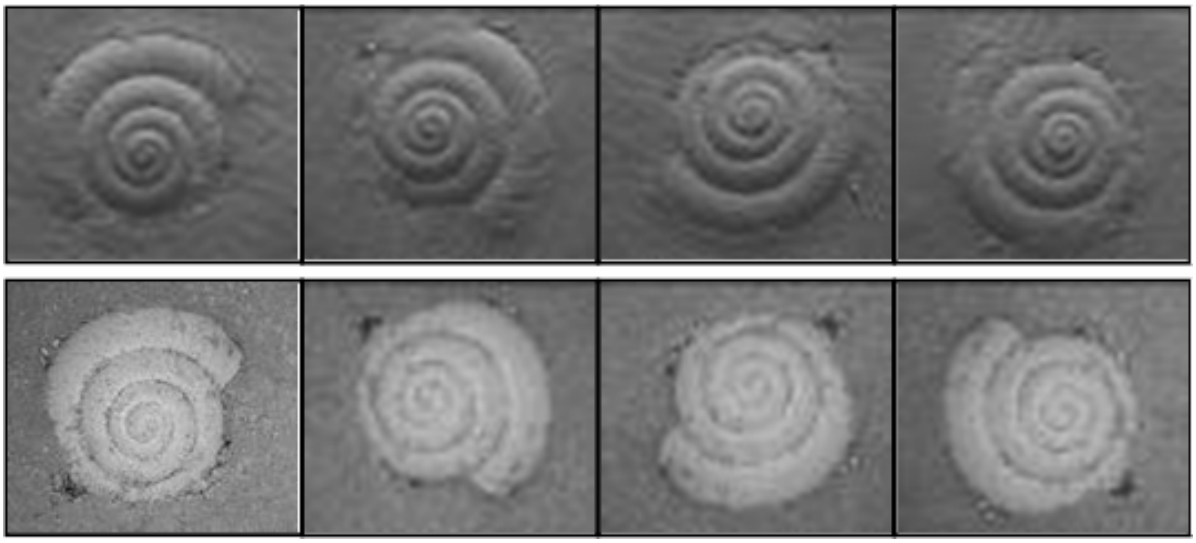


Fig. 7.14 – Stored images at the end of SEM acquisition process. BS images in the first row and IS images in the second.

The automatic alignment during the acquisition process is not really accurate for a good 3D reconstruction so we have to perform also a software image processing to refine the alignment. Then we have the four images (Fig. 7.15) that will be used by the PS algorithm to find surface heights (Fig. 7.16).

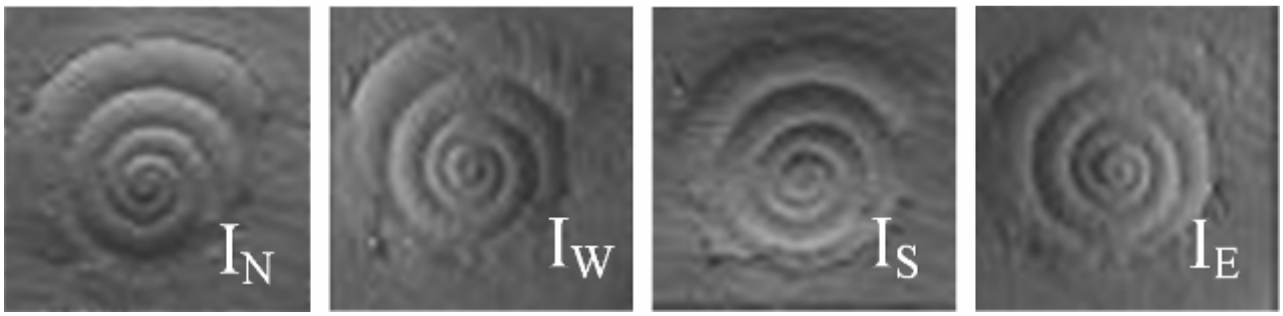


Fig. 7.15 – The images after the image processing alignment. These images are ready to be processed by PS algorithm

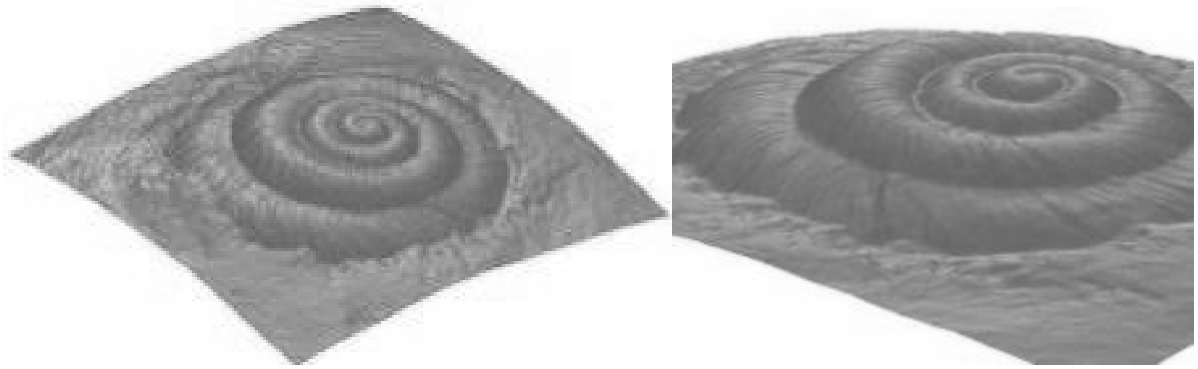


Fig. 7.16 – The 3D map of this SEM sample

Following figures shows other possible kind of SEM samples that are useful to be 3D reconstructed.

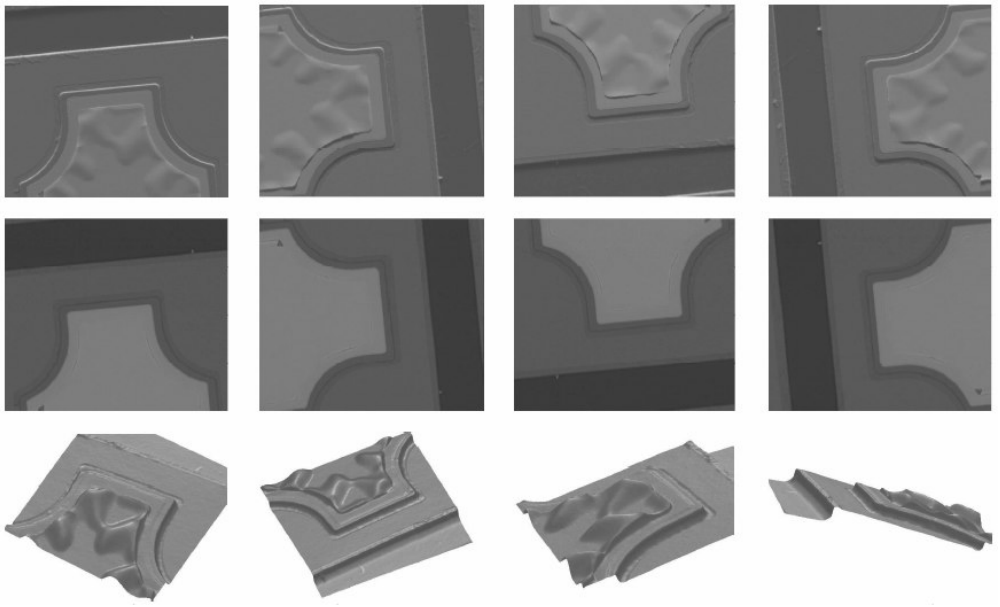
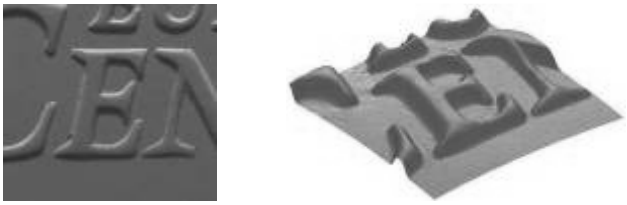


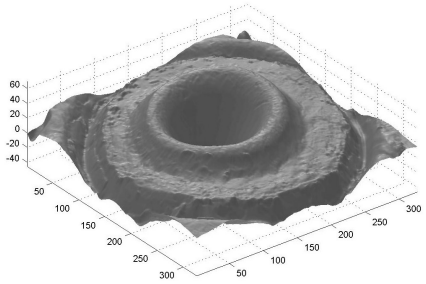
Fig. 7.17 – A set of BS images (upper row), the corresponding IS twin set (central row) and some 3D views (lower row) of the reconstructed surface of a damaged solid-state electron device.



a) b)

Fig. 7.18 – a) One of the BS SEM images of a coin detail and b) its 3D reconstruction.

a) b)



c)

Fig. 7.19 – a) Optical image and b) SEM detail of the percussion surface of an exploded bullet-case; c) reveals the 3D shape.

7.3 Polynomial Texture Map

The last application we want to describe involves an algorithm and software developed in HP labs at Palo Alto (CA). This algorithm called Polynomial Texture Mapping (PTM) takes N images of an object and interpolates this cue data to virtually render the surface as lit from any chosen directions [15]. Since we can choose the direction of illumination of a PTM sample, it's evident that we can light the surface from the directions we need to perform PS algorithm and to find surface 3D map. For this reason we download from hp-labs web site the PTM software and some samples and we find the heights of them. Next figures demonstrate the advantages of merging our and their algorithm.

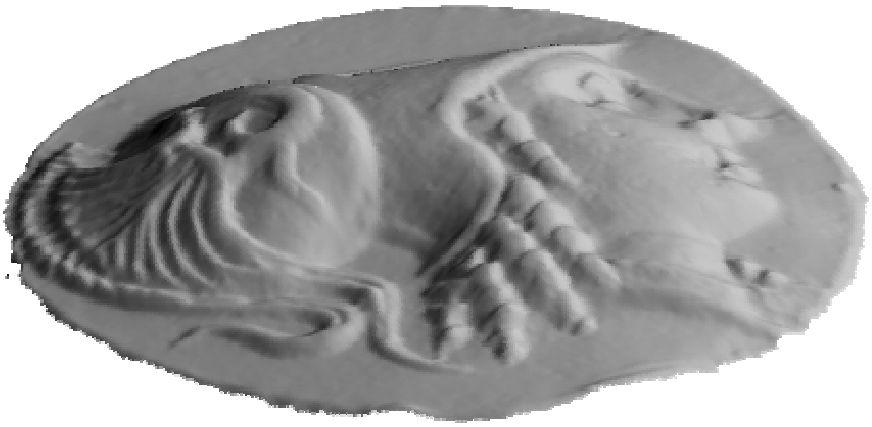
a)

b)

c)

Fig. 7.20 – a) one of PTM (Sumerian tablet) original images, b) 3D map and c) 3D map with albedo

a)

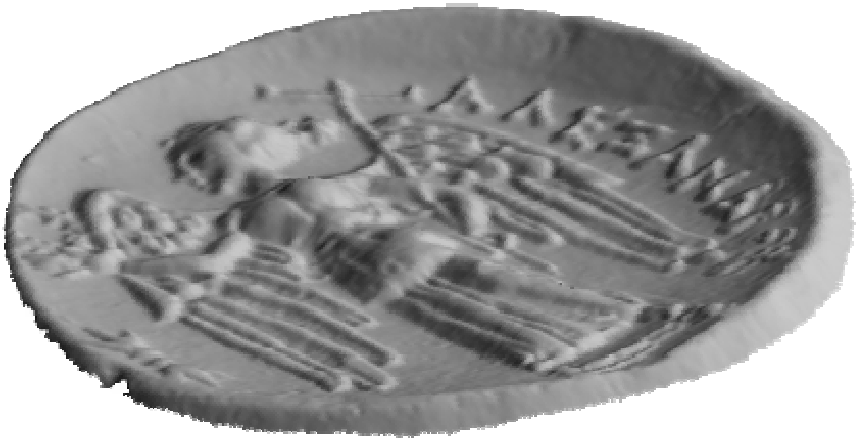


b)

c)

Fig. 7.21 – a) one of PTM (a front of an ancient coin) original images, b) 3D map and c) 3D map with albedo

a)



b)

c)

Fig. 7.22 – a) one of PTM (a rear of an ancient coin) original images, b) 3D map and c) 3D map with albedo

8. Appendix A

In order to have the best PS test images, a particular issue has to be faced: the light set-up. Since we have an analytical surface map (with real object it's more complicated and we don't deal with it here), it is necessary that the produced images set is the best to be applied to PS algorithms. In few words we should avoid in the images every non-ideal element, such as cast and self shadows, highlights and noise. First of all we should note that the smoothness of surface must be suitable for PS, which is not able enough to recognize strong surface discontinuities, and we should not have any noise or highlights; this is due to the fact that the surface is not a real object and we could choose the proper material behaviour. The last issue is the presence of shadows (cast and/or self). To solve this problem we have to analyse the given analytical surface gradient field and find a rule to decide the illumination parameters, which don't produces any shadow. Let's make an example in one dimension. Suppose we have a pyramid with the base in the x-y plane and let z-axis be its axis. Consider a section on the x-z plane (Fig. 8.1). Before moving on, note that shadows in images (both cast or self) involve the presence of a negative scalar product, between the illumination direction and surface normal, in at least one image pixel (or one object microfacet).

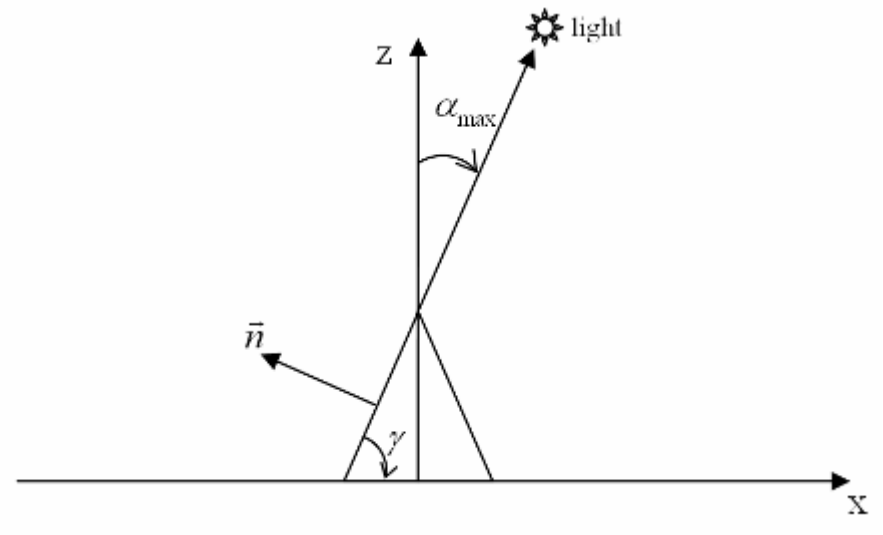


Fig. 8.1 – one dimensional geometry for best light set-up estimation

We know that the reflectance function stated that

$$R(p,q)=\frac{1+p_S p+q_S q}{\sqrt{1+p_S^2+q_S^2}\sqrt{1+p^2+q^2}}=\frac{1}{\sqrt{1+p_S^2+q_S^2}}(-p_S,-q_S,1)\cdot\frac{1}{\sqrt{1+p^2+q^2}}(-p,-q,1) \qquad (8.1)$$

where the term with the s-indices is the illumination direction vector while the other term is the surface normal. If we want to avoid shadows we should choose a proper illumination vector, which assures us that in every object microfacet the R function is non-negative. This task is really simple if we know surface gradient field. As we can easily see in one dimension, if $|\alpha| \le \frac{\pi}{2}$ (and this is a necessary condition in PS), then the maximum alpha value, to avoid shadows along the entire surface, will be

$$\alpha_{\max} = \frac{\pi}{2} - \gamma = \frac{\pi}{2} - \tan^{-1}\left(\left|\frac{\partial z}{\partial x}\right|_{\max}\right) \qquad (8.2)$$

Now let consider the usual two dimensional condition. While in the previous case we deal with alpha, since the derivative could be only in x direction, now the analysis of beta is needed. In other words we should study all possible directions of light rays to choose the maximum alpha value that does not produce any shadow independently of beta value.

In this sense, first of all note that usually the PS algorithms consider a continuous surface made by a finite number of microfacets, which begin pixels in image domain. Furthermore, we consider the pixel as an image of a square object region with unitary edge. Stated this, the depth variation (Fig. 8.2) within a pixel is

$$\delta z = p\delta x + q\delta y \qquad (8.3)$$

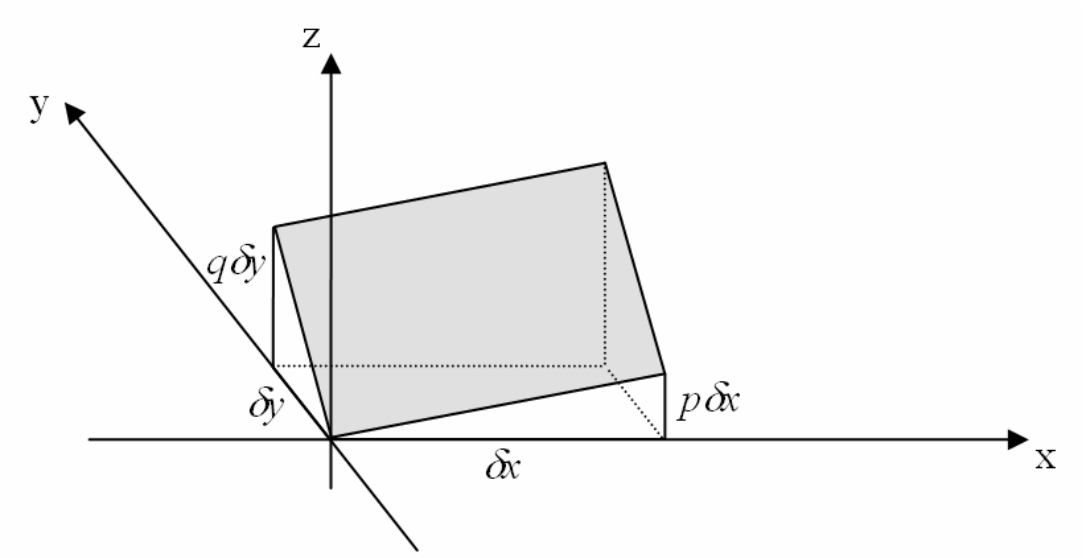


Fig. 8.2 – The change in height is the sum of the products between derivative and the step along each direction

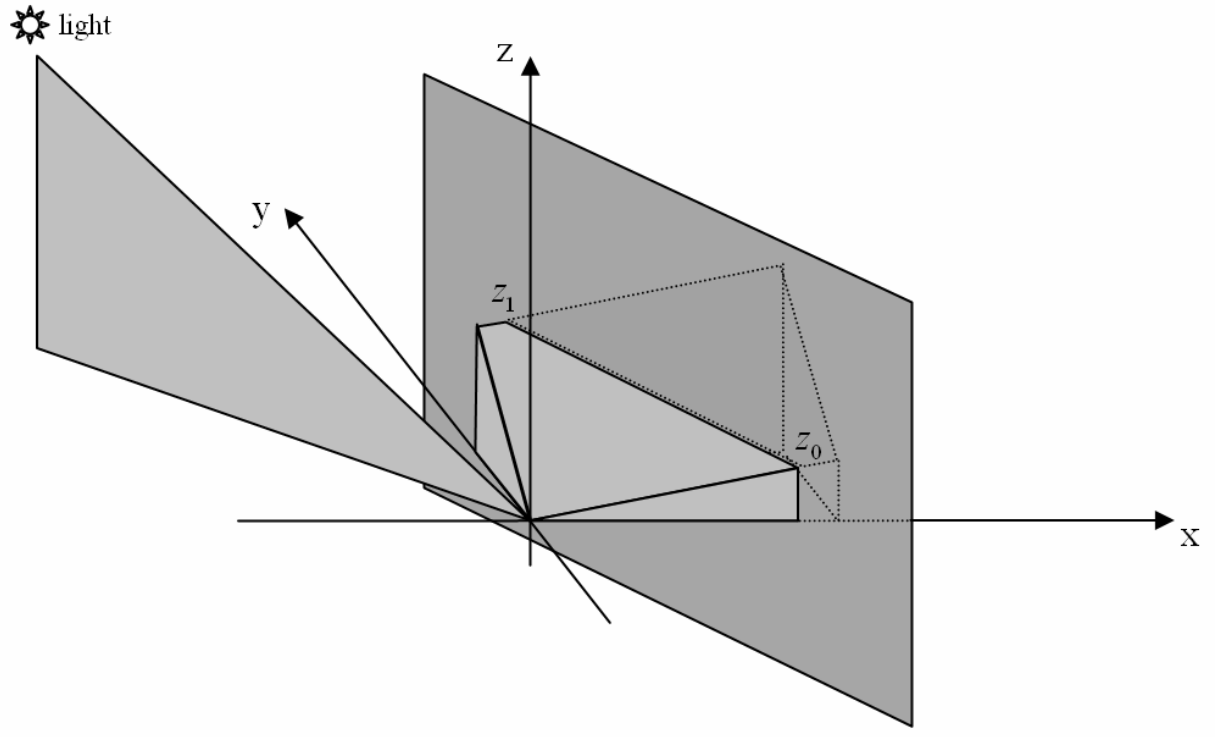


Fig. 8.3 – Image gray level value depends on the light direction vector and the normal of the $\overline{z_0 z_1}$ segment that lies on intersection plane

where δx and δy are he steps. Considering a random light direction (Fig. 8.3), the image gray level value depends on the normal of the $\overline{z_0 z_1}$ segment that lies on intersection plane. Given a range of possible (p,q) derivatives along x and y we should decide which light parameters never produce shadows. Since β might have any value, we must focus on α.

First of all we have to consider the reflectance equation in (p,q) axes and in the boundary condition $R(p,q)=0$. We can easily see that in this particular case the reflectance relation is a line with the following equation

$$q = -\frac{p_S}{q_S} p - \frac{1}{q_S} \qquad (8.4)$$

Lack of shadows holds

$$R(p,q) \geq 0 \rightarrow q \geq -\frac{p_S}{q_S} p - \frac{1}{q_S} \qquad (8.5)$$

This relation describes a part of (p,q) plane and if every (p,q) lies in this zone then we are sure that there won't be any shadows in the image. Stated this and given a range of derivatives values we could study the maximum alpha value that satisfies the condition above. Fig. 8.4 shows the geometric simple task to be solved. Here we have a region of the plane that contains any possible (p,q) couple in object surface. It's well know that the value of r depends on alpha value, while the angle between r and p (or q) axis depends on beta value. So we have to compute the maximum alpha value that permits us to cover with the circle the squared region defined by (p,q) ranges.

It's obvious that the value of $r_{\max}$ will be

$$r_{\max} = \sqrt{(\max|p|)^2 + (\max|q|)^2} \qquad (8.6)$$

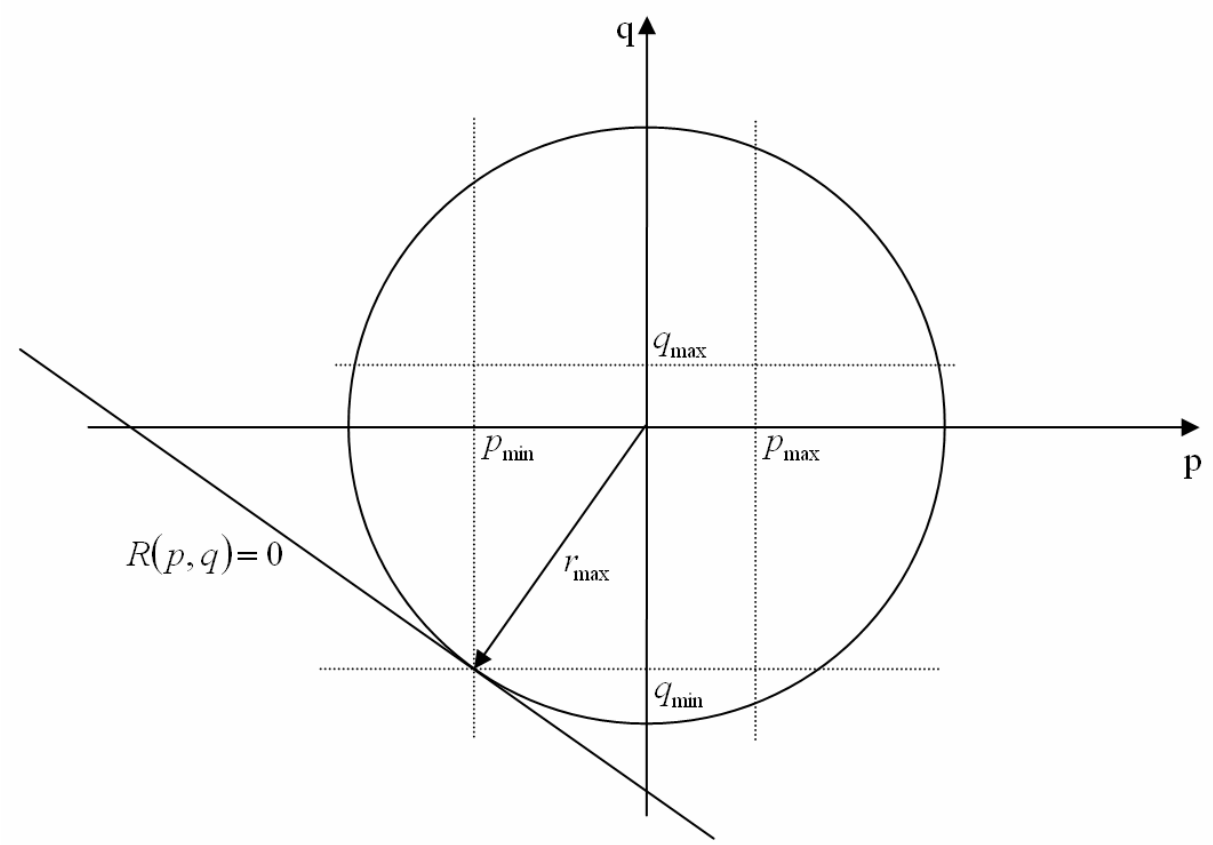


Fig. 8.4 – Once we know (p,q) range, we have to choose maximum alpha value to avoid shadows independently of beta

To compute the parameters we must consider the line equation and two constraints: first the line should be normal to $r_{\max}$ vector, second the point $(\max|p|,\max|q|)$ must lie on the line. Note that I'm not computing the parameters of the line in Fig. 8.4 but of the parallel line that is tangent to the circle in the first quadrant. It doesn't matter because the signs of the point that lie on the line are not important, since the alpha value depends only on the length of $r_{\max}$.

In mathematical terms the constraints are respectively

$$\begin{cases} \dfrac{p_S}{q_S} = \dfrac{\max|p|}{\max|q|} \\ \max|q| = -\dfrac{p_S}{q_S}\max|p| - \dfrac{1}{q_S} \end{cases} \tag{8.7}$$

This leads to

$$\begin{cases} p_S = -\dfrac{\max|p|}{(\max|p|)^2 + (\max|q|)^2} \\ q_S = -\dfrac{\max|q|}{(\max|p|)^2 + (\max|q|)^2} \end{cases} \tag{8.8}$$

Considering that (α, β) define the light position vector and that we don't care about beta we could find with no trouble

$$\alpha_{\max} = \arccos\left(\frac{(\max|p|)^2 + (\max|q|)^2}{\sqrt{(\max|p|)^2 + (\max|q|)^2 + \left[(\max|p|)^2 + (\max|q|)^2\right]^2}} \right) \tag{8.9}$$

Finally, if we choose $\alpha \leq \alpha_{\max}$, we will sure that no shadow could affect our 3D reconstruction, independently of beta.

Generally we won't have a perfect squared region which identifies the gradient range. So, after plotting all surface (p, q) pairs, we should find the pair that is the most distant one from axis intersection. The p and q values of this point become the maximum values in (8.9).

9. References

[1] B. K. P. Horn, “Robot Vision”, The MIT Press.

[2] Diego Nehab, Szymon Rusinkiewicz, James Davis, Ravi Ramamoorthi, “Efficiently Combining Positions and Normals for Precise 3D Geometry”, ACM Transactions on Graphics (Proc. SIGGRAPH), August 2005

[3] R.J. Woodham, “Photometric Method for Determining Surface Orientation from Multiple Images,” Optical Eng., vol. 19, no. 1, pp. 139-144, 1980.

[4] R.J. Woodham, “Gradient and Curvature from the Photometric-Stereo Method, Including Local Confidence Estimation,” J. Optical Soc. Am., vol. 11, no. 11, pp. 3050-3068, Nov. 1994.

[5] R.J. Woodham, Y. Iwahori, and R.A. Barman, “Photometric Stereo:Lambertian Reflectance and Light Sources with Unknown Direction and Strength,” technical report, Dept. of Computing Science, Univ. of British Columbia, 1991.

[6] S.K. Nayar, K. Ikeuchi, and T. Kanade, “Determining Shape and Reflectance of Hybrid Surfaces by Photometric Sampling,” IEEE Trans. Robotics and Automation, vol. 6, no. 4, pp. 418-431, Aug. 1990.

[7] H.D. Tagare and R.J.P. deFigueiredo, “A Theory of Photometric Stereo for a Class of Diffuse Non-Lambertian Surfaces,” IEEE Trans. Pattern Analysis Machine Intelligence, vol. 13, no. 2, pp. 133-152, Feb. 1991.

[8] E.N. Coleman and R. Jain, “Obtaining 3-Dimensional Shape of Textured and Specular Surfaces Using Four-Source Photometry,” Computer Graphics and Image Processing, vol. 18, pp. 309-328, 1982.

[9] P.H. Christensen and L.G. Shapiro, “Three-Dimensional Shape from Color Photometric Stereo,” Int’l J. Computer Vision, vol. 13, no. 2, pp. 213-227, 1994.

[10] G.J. Klinker, A Physical Approach to Color Image Understanding. A.K. Peters, 1993.

[11] http://www.irit.fr/~Frederic.Courteille/index.php

[12] Svetlana Barsky and Maria Petrou, “The 4-Source Photometric Stereo Technique for Three-Dimensional Surfaces in the Presence of Highlights and Shadows” IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 25, no. 10, October 2003

[13] Frankot R.T. and Chellappa R., “A Method for Enforcing Integrability in Shape from Shading Algorithms”, PAMI, Vol 10, No. 4, pp.439-451, 1988.

[14] http://www.nrbook.com/a/bookcpdf.php

[15] Malzbender, T., Gelb, D., Wolters, H., “Polynomial Texture Maps”, Computer Graphics (SIGGRAPH 01 PROCEEDINGS), August 2001.