# CS6320: 3D Computer Vision
# Project 3
# Shape from Shading

Arthur Coste: *coste.arthur@gmail.com*

March 2013

# Contents

## 1 Introduction

In this third project we are going to extend the analysis we started in the previous project dealing with depth and geometry of objects in a scene. Indeed, knowing what is in a scene acquired with cameras is an important topic with many applications. As we mentioned in the previous project these techniques require at least two images to be able to reconstruct depth or disparity. As we are going to see in this new project, we will focus on the use of light to reconstruct the geometry of an object and this technique will also require multiple images to provide results.

In this report, we will first discuss some theoretical problems related to Shape from Shading properties such as Lambertian surfaces or Reflectance properties and a complement on the stereo model dealing with disparity accuracy in measurements.

In this project, the implementation is made using MATLAB, the functions we developed are included with this report and their implementation presented in this document.

The following functions are associated with this work :

- $photometric()$ Stand alone function
- $test\_fourier()$ Stand alone test function

Note : All the images and drawings were realized for this project so there is no Copyright infringement in this work.

## 2 Theoretical Problems

### 2.1 Bidirectional Reflectance Function

In this part, we are going to discuss some aspects dealing with the Reflectance model. Here is a drawing setting up the situation:

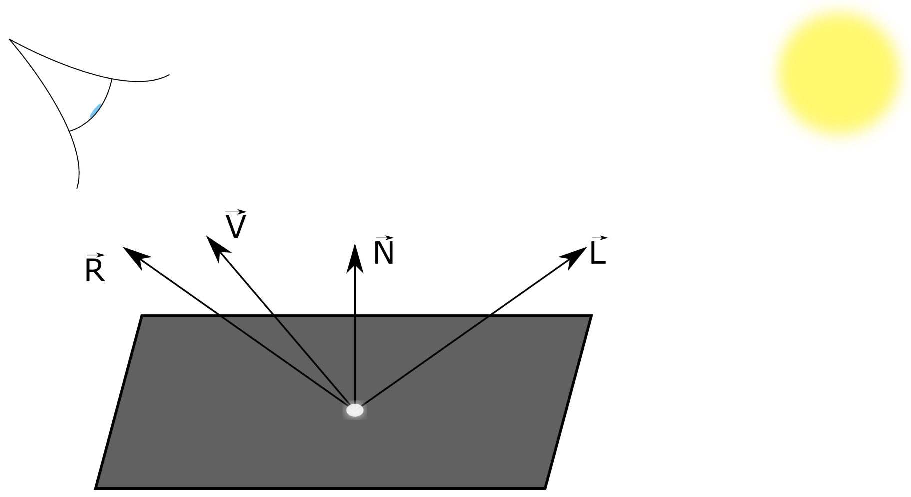

Figure 1: Set up

In the previous set up we have a surface patch that receive light and which may emit some of it according to the material properties. So, we have a light source, which is defined using the $\overrightarrow{L}$ vector, we have a normal to the surface $\overrightarrow{N}$ and we also have the observer which is defined by $\overrightarrow{V}$. In this case the observer will be more likely a camera than a human eye. We can add to this model a possible reflection $\overrightarrow{R}$ of the incident light which is defined according to Snell-Descartes laws with the same angle with the surface as the light source and which would generally be different from $\overrightarrow{V}$.

#### 2.1.1 Specular versus Diffuse light

In this first question, we are going to discuss the differences between the specular light which create specularities and the diffuse reflection.

Firstly let's define briefly what is specular light and diffuse light. Specular light is a bright spot of light which is visible on certain objects when they are illuminated. This aspect depends of course of the material physical properties, some are more likely to produce specular spots than other materials. Specular spots are directly linked to Snell-Descartes reflection ray.
Secondly the diffuse component is linked to reflection of the incident light in many directions by the surface of the object.
Let's now represent in a drawing these two components of light reflection on a surface.

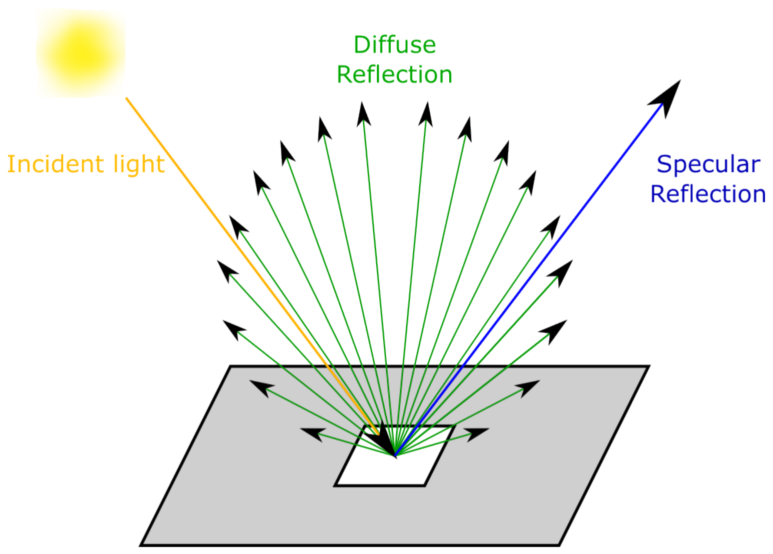

Figure 2: Diffuse and specular light created by a surface element

So based on the previous drawing, we can explain that specularities are brighter than diffuse reflection because diffuse reflection as its name states is diffusing everywhere, while on the other hand the specular reflection is highly directed in the symmetric direction of the light source with respect to the surface normal.
So if we consider the same amount of light energy the observer gets a lot more from specularities than from diffuse reflection.

Finally, let's illustrate this statement with two computer generated images to prove this aspect.

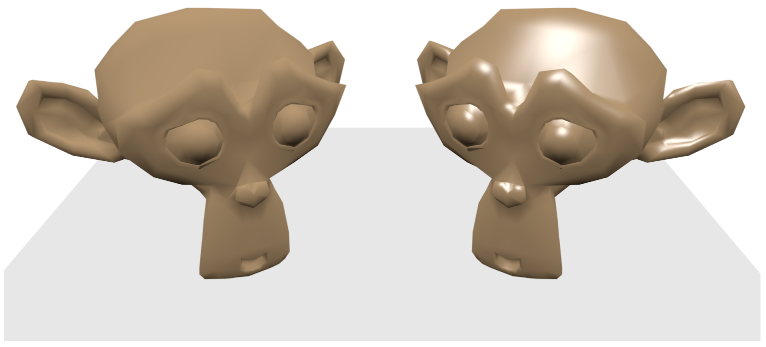

Figure 3: On the left only diffuse light, on the right diffuse light and specular reflection

### 2.1.2 Lambertian Surface

In this second question we are going to discuss one of the property of Lambertian surfaces.

Let's start by defining briefly what is a Lambertian surface. A Lambertian surface is a surface which as the property of invariant luminance according to the viewing angle. Let's add that changing the observation angle is the same thing as rotating the surface, because what is at stake here is the angle between the surface normal and the observation view point.
We are now going to illustrate how is this property true. Let's consider a slightly different situation as the one presented above with the reflection to illustrate this aspect.

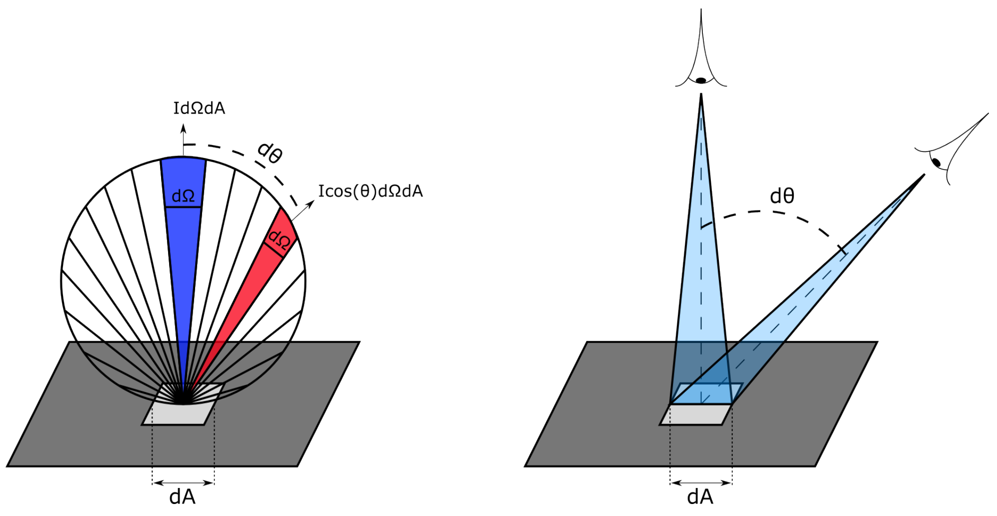

Figure 4: Illustration of Lambertian surface properties

Our general equation providing luminance is the following:

$$L = \rho \, \langle \overrightarrow{n_s}, \overrightarrow{n} \rangle \tag{1}$$

Where $\rho$ is the albedo and $\langle \cdot, \cdot \rangle$ is the dot product (scalar product). In our drawing, the albedo $\rho$ is defined by : $\rho = Id\Omega dA$. Indeed, it makes sense to have an albedo function of the surface patch dimension and the diameter of the observing cone. In this case we do a 2D projected reasoning to simplify. So we have the following luminance equation, which is our light flux in photon emitted per second for each location :

$$L_{\overrightarrow{n}} = Id\Omega dA \tag{2}$$

$$L_{cos(\theta)} = Icos(\theta)d\Omega dA \tag{3}$$

If we now compute the radiance which is the light flux computed for the solid angle subtended by the observation point we end up with:

$$I_{\overrightarrow{n}} = \frac{Id\Omega dA}{d\Omega_0 dA_0} \tag{4}$$

$$I_{cos(\theta)} = \frac{Icos(\theta)d\Omega dA}{d\Omega_0 cos(\theta)dA_0} = \frac{Id\Omega dA}{d\Omega_0 dA_0} = I_{\overrightarrow{n}} \tag{5}$$

So we briefly proved with those equations that the radiance is the same, to make a better proof of that we should have set up a better physical model and set up.
The conclusion of that part would be to say that, the change of view point with an angle $\theta$ regarding the surface normal is of course affecting the luminance, but the eye being sensitive to radiance, this shift will be carried into radiance and will affect the same way the solid angle resulting from observation causing the simplification of $cos(\theta)$ and providing at the end the Lambertian property in terms of radiance for the human eye.

## 2.2 Reflectance Map

After discussing the Lambertian property of some surfaces, we are going in this section to discuss the case of general surfaces that cannot fall into the Lambertian assumption. In this case we are going to work on the case of increasing brightness regarding the observation angle. The practical application of this phenomenon deals with the surface of the moon. In the previous question, we presented and illustrated the special case of Lambertian surfaces where luminance was independent from the viewing angle, in this question we are going to deal with a more general case where the Lambertian approximation does not hold.

Let's firstly set up the situation:

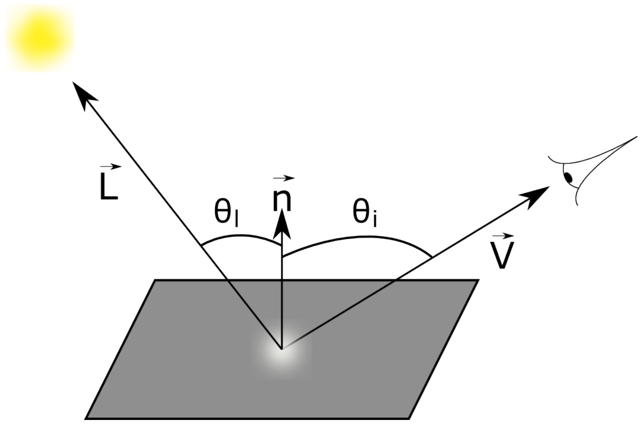

Figure 5: Set up of the situation

In this set up, we can easily locate the two angles $\theta_i$ between the surface normal and the light source, and $\theta_l$ between the surface normal and the observation point. In the previous Lambertian case, we assumed the following equation to get the luminance :

$$L \propto \rho \langle \overrightarrow{n_s}, \overrightarrow{n} \rangle \tag{6}$$

This equation was done under the influence of the hypothesis that the angle $\theta_l$ was cancelling out.

But in this more general case we cannot cancel it out so we have the following equation:

$$L \propto \rho cos(\theta_i) \times \frac{1}{cos(\theta_l)} \tag{7}$$

Using the drawing above we can rewrite this equation as follow:

$$L \propto \rho \frac{\langle \overrightarrow{n_s}, \overrightarrow{n} \rangle}{\langle \overrightarrow{n_l}, \overrightarrow{n} \rangle} \tag{8}$$

Let's now express separately the two scalar products reminding that:

$$\left\langle \overrightarrow{A}, \overrightarrow{B} \right\rangle = ||\overrightarrow{A}|| \times ||\overrightarrow{B}|| \times cos(\widehat{\overrightarrow{A}, \overrightarrow{B}}) \qquad \Rightarrow \qquad cos(\widehat{\overrightarrow{A}, \overrightarrow{B}}) = \frac{\left\langle \overrightarrow{A}, \overrightarrow{B} \right\rangle}{||\overrightarrow{A}|| \times ||\overrightarrow{B}||} \tag{9}$$

$$cos(\theta_i) = \frac{\langle \overrightarrow{n_s}, \overrightarrow{n} \rangle}{||\overrightarrow{n_s}||||\overrightarrow{n}||} = \frac{\begin{pmatrix} -p_s \\ -q_s \\ 1 \end{pmatrix} \begin{pmatrix} -p \\ -q \\ 1 \end{pmatrix}}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}} = \frac{p_s \times p + q_s \times q + 1}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}} \tag{10}$$

$$cos(\theta_l) = \frac{\langle \overrightarrow{n_l}, \overrightarrow{n} \rangle}{||\overrightarrow{n_l}||||\overrightarrow{n}||} = \frac{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} -p \\ -q \\ 1 \end{pmatrix}}{\sqrt{1}\sqrt{p^2 + q^2 + 1}} = \frac{1}{\sqrt{p^2 + q^2 + 1}} \tag{11}$$

So by aggregating all of the previous we finally obtain:

$$L \propto \rho cos(\theta_i) \times \frac{1}{cos(\theta_l)} \qquad \Rightarrow \qquad L \propto \rho \frac{p_s p + q_s q + 1}{\sqrt{p_s^2 + q_s^2 + 1}} \tag{12}$$

Let's now determine the isophote set which is the set of curves of constant brightness by solving the following equation:

$$L = cste = R(p, q) \qquad \Rightarrow \qquad \rho \frac{p_s p + q_s q + 1}{\sqrt{p_s^2 + q_s^2 + 1}} = cste \tag{13}$$

Due to the fact that $p_s$ and $q_s$ refers to the light source position which is fixed for a given picture we can simplify a lot this equation:

$$p_s p + q_s q = cste \times \frac{\sqrt{p_s^2 + q_s^2 + 1}}{\rho} - 1 \qquad \Rightarrow \qquad p_s p + q_s q = cste_2 \tag{14}$$

This is a line equation in terms of $p$ and $q$:

$$q = \frac{cste_2}{q_s} - \frac{p_s}{q_s} p \tag{15}$$

So for a sphere and a given light source position, the set of isophote curve is a set of parallel lines.

To be able to determine the values of $p$ and $q$ needed to reconstruct the local normal we will need at least to images, to obtain an interception point between the two sets of lines such as represented in the following figure:

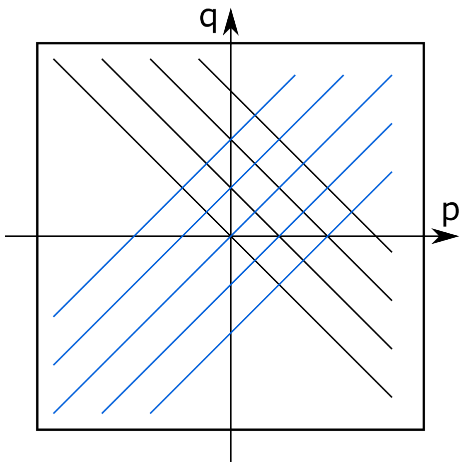

Figure 6: Two sets of isophote lines associated to two specific light source positions

## 2.3 Stereo System

Let's remind the set up and the disparity equation from our stereo system :

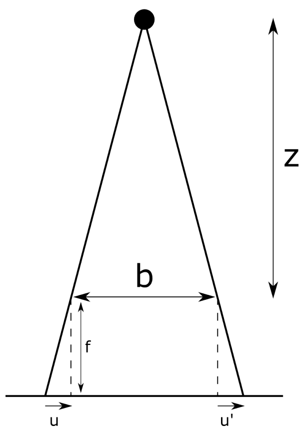

Figure 7: Rectified stereo set up

$$d = \frac{-Bf}{Z} \tag{16}$$

If we derive the previous disparity equation with respect to $Z$ we obtain:

$$\frac{\partial d}{\partial Z} = \frac{Bf}{Z^2} \tag{17}$$

If we analyse this equation in terms of variation of depth $Z$, we can say that the disparity evolves as $d \propto f(\frac{1}{Z^2})$

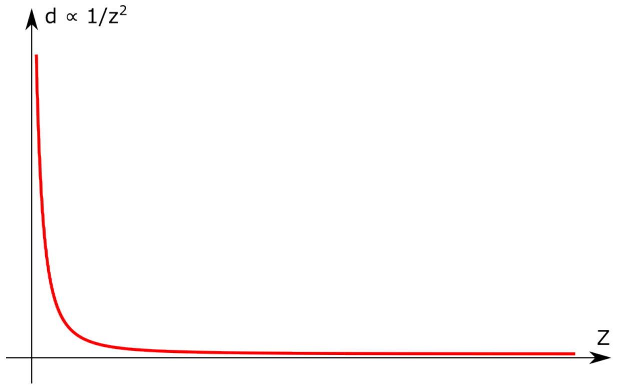

Figure 8: Evolution of disparity according to depth

So according to this equation, if objects are far away it means that their depth $(Z)$is large so the disparity is small and respectively if they are close to the observer the depth $(Z)$ is small so the disparity is larger. Let's explain how this makes sense. The first very easy trick to understand it is to use our eyes which are an amazing stereo system. An easy set up is to face a window, take a pen close to your eyes and close one eye locate the position of the pen with respect to the environment and take another landmark far away. Them close this eye and open the other. Relatively speaking, the pen should have moved a lot with respect to the environment while your landmark far away would seem to have a smaller shift.
This aspect was also presented in the last assignment results, and maybe unfortunately not explained clearly at that time. But we can show again the result of disparity computation from a stereo pair.

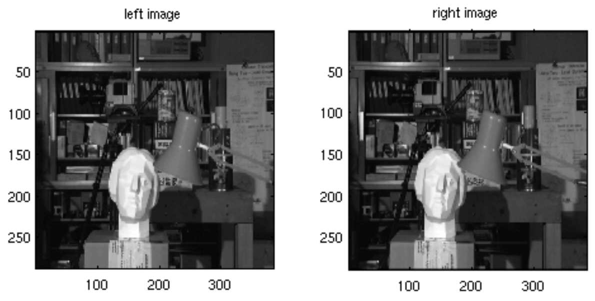

Figure 9: Input stereo pair

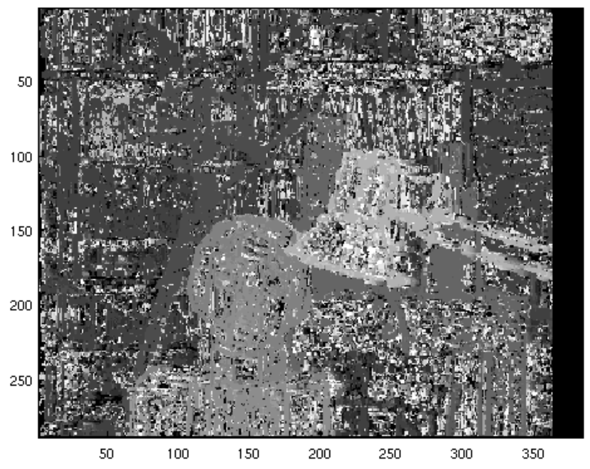

Figure 10: Disparity map associated to the previous rectified stereo pair

On the previous disparity map, the color encoding it is the bigger the disparity the brighter, so we can clearly get to the same conclusion as before, where close objects have a more important disparity than objects with a greater distance to the observer (depth).

### 2.3.1 Accuracy of stereo model

Knowing the accuracy of a vision system is important when we start doing measurement on the acquired images this is why in this section, we are going to discuss the possible dependence of the error in depth estimation as a function of the baseline and the focal length. Do do so, let's compute all the partial derivatives of our depth function:

$$Z = \frac{-Bf}{d} \tag{18}$$

$$\frac{\partial Z}{\partial d} = \frac{Bf}{d^2} \tag{19}$$

$$\frac{\partial Z}{\partial B} = \frac{-f}{d} \tag{20}$$

$$\frac{\partial Z}{\partial f} = \frac{-B}{d} \tag{21}$$

Then, let's compute the error using the error estimation formula on Z.

$$\partial Z = \sqrt{(\frac{\partial Z}{\partial d}\delta d)^2 + (\frac{\partial Z}{\partial B}\delta B)^2 + (\frac{\partial Z}{\partial f}\delta f)^2} \tag{22}$$

$$\partial Z = \sqrt{(\frac{Bf}{d^2}\delta d)^2 + (\frac{-f}{d}\delta B)^2 + (\frac{-B}{d}\delta f)^2} \tag{23}$$

Let's now assume that the values of the focal distance $f$ and the baseline $B$ are exactly known. It implies that there is no uncertainty on their values so that $\delta B = \delta f = 0$ so we can simplify our equation to:

$$\partial Z = \sqrt{(\frac{Bf}{d^2}\delta d)^2} \qquad \Rightarrow \qquad \partial Z = \frac{Bf}{d^2}\delta d \tag{24}$$

If we plug in the equation of disparity we end up with :

$$\frac{\partial Z}{Z} = \frac{\partial d}{d} \tag{25}$$

Here is another way to get this relationship without using the given formula and by doing differentiation in case of uncertainty measurements. Let's take the logarithm of the disparity equation

$$log(Z) = -log(b) - log(f) + log(d) \tag{26}$$

and if we derive it and take variation maximization instead of derivatives we end up with:

$$\frac{dZ}{Z} = -\frac{dB}{B} - \frac{df}{f} + \frac{dd}{d} \qquad \Rightarrow \qquad \frac{\Delta Z}{Z} = \frac{\Delta B}{B} + \frac{\Delta f}{f} + \frac{\Delta d}{d} \tag{27}$$

which leads to the previous result in case of exact values of $B$ and $f$. In this case our $\frac{\Delta Z}{Z}$ is exactly the same thing as $|\frac{\delta Z}{Z}|$.
In case of uncertainty on the values of all the variables, they modify the accuracy of depth reconstruction as a ratio proportional to the measured depth meaning that the bigger these uncertainties get the more it will influence our estimation of depth.

If we want to obtain a ratio smaller than 1% we have:

$$\frac{\Delta Z}{Z} < \frac{1}{100} \qquad \Rightarrow \qquad \frac{\Delta d}{d} < \frac{1}{100} \tag{28}$$

This implies that the uncertainty on the measure of disparity has to be really small to have an accurate measurement of depth.

## 3 Practical Problems

### 3.1 Photometric Stereo and Shape from Shading

In this section, we are going to discuss and implement the shape from shading technique. This technique is aiming at estimating and reconstructing the geometry of an object using multiple view acquisition of the object with a different light position. By moving the light source we are going to create a modification of the shading on the acquired image which will allow us to compute the surface normal for a given patch. Once we have a set of normal vector describing the whole object, we can estimate the possible geometry of the object. In this section we are going to present the theory supporting this technique and then present our implementation and results firstly on synthetic data set and secondly on images of real objects.

#### 3.1.1 Theoretical presentation

Let's considerate the following set up for our experiment:

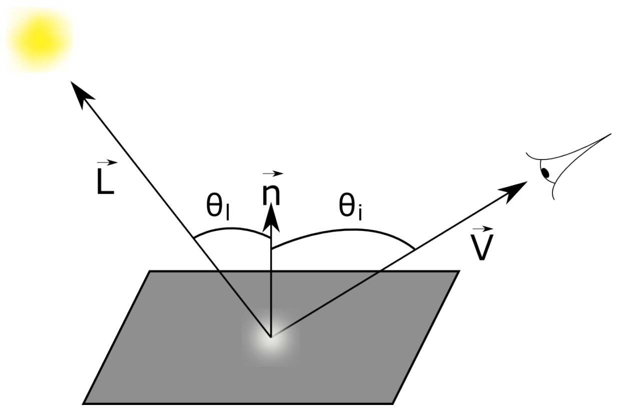

Figure 11: Original set up

Thanks to the previous theoretical discussion about Lambertian surface, we are going to assume the Lambertian surface hypothesis for our practical implementation and study. So we can quickly mention what was discussed above:

$$L = \rho\frac{cos(\theta_i)}{cos(\theta_l)} \qquad \text{is simplified to} \qquad L = \rho cos(\theta_i) \tag{29}$$

The goal of this method is to be able to reconstruct the shape of the object according to the intensity which is a function of the shading on the image. So the first step to be able to reconstruct the shape is to estimate the normal vector to a surface element. Indeed, by computing the normal to a surface we can then by knowing that the normal is orthogonal to the surface reconstruct the shape.
Let's firstly present the general method to estimate the surface normal.

A general surface can be expressed as a 2D height function, where the height is function of the location on the plan such as:

$$\begin{cases} \mathbb{R}^2 \to \mathbb{R} \\ (x,y) \mapsto S = f(x,y) = z(x,y) \end{cases} \tag{30}$$

So we can represent each surface element as a vector:

$$S = \begin{pmatrix} x \\ y \\ z(x,y) \end{pmatrix} \tag{31}$$

Then we can compute the partial derivative of $z$ according to both $x$ and $y$.

$$\frac{\partial S}{\partial x} = \begin{pmatrix} 1 \\ 0 \\ \frac{\partial z(x,y)}{\partial x} \end{pmatrix} \quad , \quad \frac{\partial S}{\partial y} = \begin{pmatrix} 0 \\ 1 \\ \frac{\partial z(x,y)}{\partial y} \end{pmatrix} \tag{32}$$

We can then define two variables $p$ and $q$ such as:

$$\begin{cases} p = \frac{\partial z(x,y)}{\partial x} \\ q = \frac{\partial z(x,y)}{\partial y} \end{cases} \tag{33}$$

So we obtain:

$$\frac{\partial S}{\partial x} = \begin{pmatrix} 1 \\ 0 \\ p \end{pmatrix} \quad , \quad \frac{\partial S}{\partial y} = \begin{pmatrix} 0 \\ 1 \\ q \end{pmatrix} \tag{34}$$

Then to compute the normal vector, we are going to rely on the fact that a normal is always orthogonal to the surface so it's always orthogonal to the surface gradients. The best way to obtain an orthogonal vector from two vectors already orthogonal is to compute the cross (vector) product:

$$\overrightarrow{n} = \frac{\partial S}{\partial x} \times \frac{\partial S}{\partial x} = \begin{pmatrix} 1 \\ 0 \\ p \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ q \end{pmatrix} = \begin{pmatrix} -p \\ -q \\ 1 \end{pmatrix} \tag{35}$$

The last step to get a real normal vector is to give it a unit norm. So finally we have:

$$\hat{\overrightarrow{n}} = \frac{\overrightarrow{n}}{\sqrt{p^2 + q^2 + 1}} \tag{36}$$

So we obtained the equation to estimate the normal vector associated to a surface element, which is function of the image intensity gradients regarding the 2D spatial variables of the image: $x$ and $y$. The unknown to determine in this problem are then $p$ and $q$. So let's discuss what they represent and how we can get them.

In our problem, we deal with a light source which also have its own normal vector and which could be expressed using the same method:

$$\hat{\overrightarrow{n_s}} = \frac{\overrightarrow{n_s}}{\sqrt{p_s^2 + q_s^2 + 1}} \qquad \text{with} \qquad n_s = \begin{pmatrix} -p_s \\ -q_s \\ 1 \end{pmatrix} \tag{37}$$

If we get back to the luminance equation presented before and plug the equation associated to our normal vector we obtain:

$$L = \rho cos(\theta_i) = \rho \langle \overrightarrow{n_s}, \overrightarrow{n} \rangle = \rho \frac{\overrightarrow{n_s}\overrightarrow{n}}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}} = \frac{pp_s + qq_s + 1}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}} \tag{38}$$

This equation is a function of variables $p$ and $q$ which is the reflectance: $R(p,q)$. We mentioned it before in the theoretical problems section. Here again we can then determine the contours of constant reflectance called isophote, by solving:

$$L = cste \qquad \Rightarrow \qquad \rho \frac{\overrightarrow{n_s}\overrightarrow{n}}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}} = \frac{pp_s + qq_s + 1}{\sqrt{p_s^2 + q_s^2 + 1}\sqrt{p^2 + q^2 + 1}} = cste \tag{39}$$

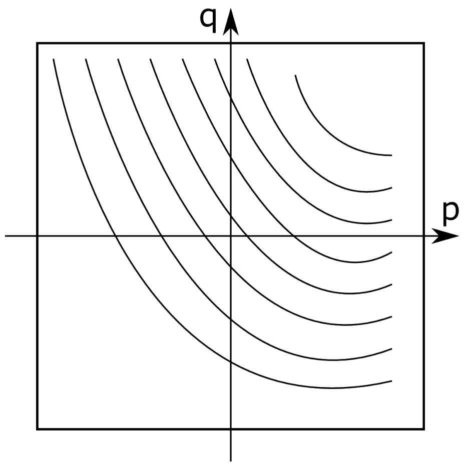

Figure 12: Set of isophote curves associated to a specific light source position

In the general case if we modify the previous equation, we can end up with the general equation of a conic curve. Under the assumption that the light source is behind the camera we can simplify the previous equation and the isophote set become a set of concentric circles.

We now have a map of curves defined with our two unknowns $p$ and $q$. But given a curve we can never get the associated values of $p$ and $q$ because there is a infinite set of possibles solutions as we deal with a continuous curve. We need more information to extract a single value. To do so, we are going to use a technique that reminds a lot triangulation problems. Indeed, one important thing to add is that we assume known the light source position meaning that $p$ and $q$ are the real unknown values of the problem. So if we acquire a second picture, we can using the previous method compute the associated set of isophote. If we plot them on the same map as the previous set, curves are going to intercept each other at various locations. This is an interesting step because we reduce a large number of possible values associated to a curve, to a set of two possible points.

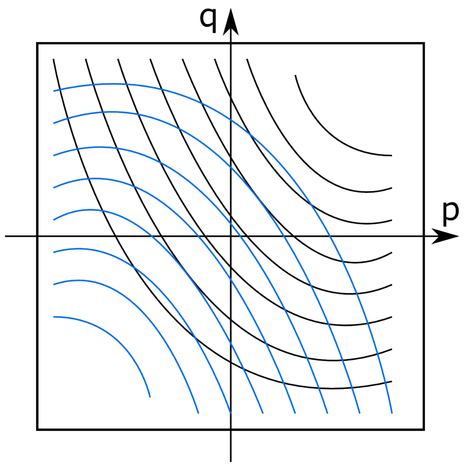

Figure 13: Two sets of isophote curves associated to two specific light source positions

But that's not enough, so this is why a third image will be necessary to reduce the possible solutions to a single point.

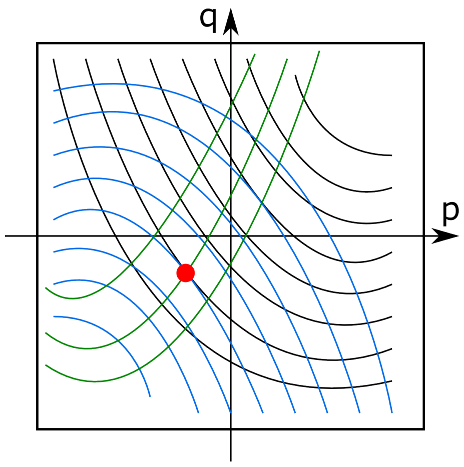

Figure 14: Three sets of isophote curves associated to three specific light source positions

So theoretically, with only 3 images we are able to get the values of $p$ and $q$. But in practical situations its common that there is noise and sources of uncertainty in computation so using more images could enforce the estimation of our parameters.

So for a given surface element, using three images we have the following system of equations:

$$\begin{cases} L_1 = \rho S_1 \overrightarrow{n} \\ L_2 = \rho S_2 \overrightarrow{n} \\ L_3 = \rho S_3 \overrightarrow{n} \end{cases} \qquad \text{with} \qquad \overrightarrow{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \tag{40}$$

Here the albedo $\rho$ is considered constant.
So, in this theoretical case, we have to solve a three equation system with three unknowns, which can be performed using Gauss Jordan elimination. With more images, we obtain an overdetermined system with more equations than unknowns so we need to use a Least Square Estimate for our parameters.

A constant albedo as mentioned before is not the general case and the albedo is also a function of the spatial position on the image: we have $\rho(x, y)$. So in this case we have the following equation:

$$I(x, y) = \kappa\rho(x, y)\overrightarrow{n}(x, y)\overrightarrow{S} \qquad \Rightarrow \qquad I(x, y) = g(x, y)V_i \tag{41}$$

So this can lead us to compute $g(x, y)$ which embed in its norm the albedo value and if normalized provides the normal surface vector orientation:

$$\begin{cases} \rho = ||\hat{g}|| \\ \overrightarrow{n} = \frac{\hat{g}}{||\hat{g}||} \end{cases} \tag{42}$$

So, finally we obtain the estimation of the albedo and the normal vector to the surface.

The last thing to do is to integrate all the normal vectors to reconstruct the surface. This is again a tricky part because we need to enforce the main characteristic of a surface which is to be $\mathcal{C}^2$ *i.e* continuous and integrable. Because objects are usually made with continuous surfaces. The standard integration algorithm will be presented in the implementation part because it's what we used, but other more sophisticated method such as Horn method based on Fourier Transform, Shah approach or some others.

### 3.1.2 Results and discussion

Here are some results of our implementation of the shape from shading method. In a first time we are going to work with synthetic data set and then we will use it on real data sets. For each data set, we will present the estimated albedo map, which comes from the norm of our $g$ vector and then if possible a needle diagram representing the orientation of the local surface normal vectors or a decomposition of it on three images.

The first test image we are going to use is composed of the following 4 images.

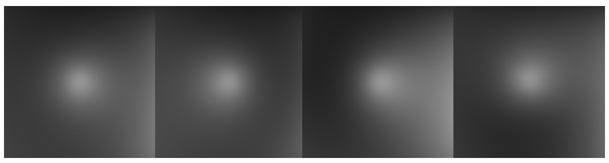

Figure 15: Four images with different light source position used to reconstruct shape

According to the images it seems that we are dealing with a bright area in the center and darker surrounding areas. The first thing to do is to reconstruct the albedo map and estimate the local normal vectors. To compute them, we used the method presented before with our implementation presented in the implementation section.

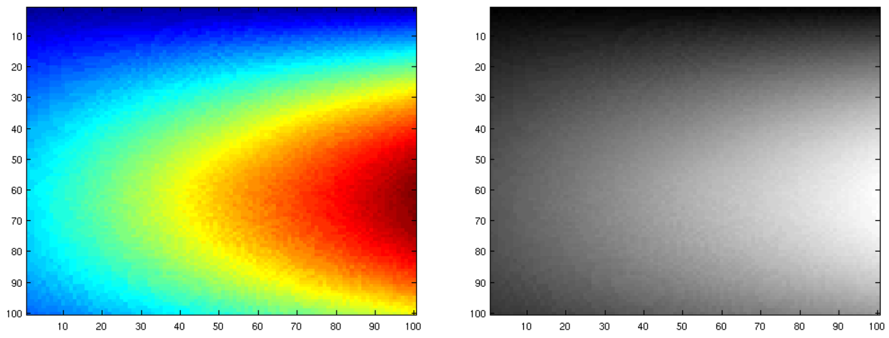

Figure 16: Computed albedo map

Let's analyse this result because it's quite surprising at first sight regarding the images we used. If we look at the second and the third image where the light source is symmetrical. We have $(0.2, 0, 1)$ and $(-0.2, 0, 1)$ for the second and the third image respectively. If we compare them we can see that the third image is brighter on the right side of the third image. We can also add that this albedo map make some sense with regard to the dark areas which appear to have a low albedo value and the center of the image has an albedo that seems to agree with the images. Let's now analyse the normal vector result as a needle map first:

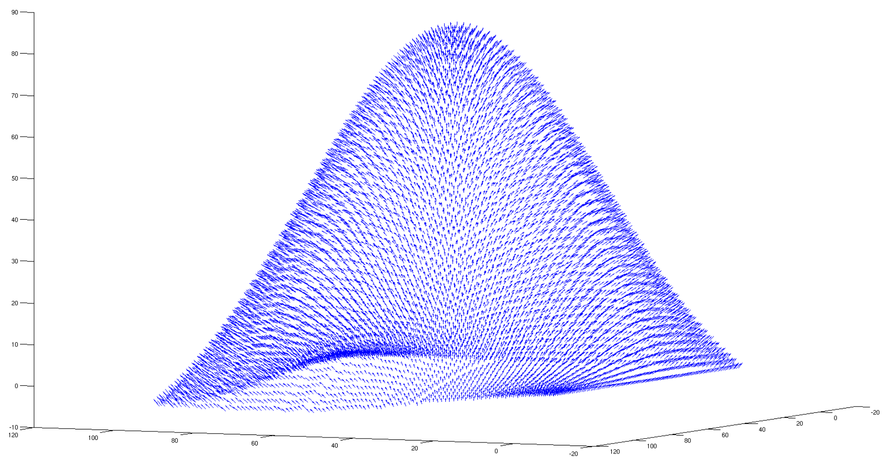

Figure 17: Computed normal vector needle map

If we analyse this result, we can clearly see the shape of a kind of bi dimensional Gaussian kernel. Then the brightest area on the pictures is the center of the image which clearly correspond to the summit of this shape and the normals are clearly oriented to the light source.

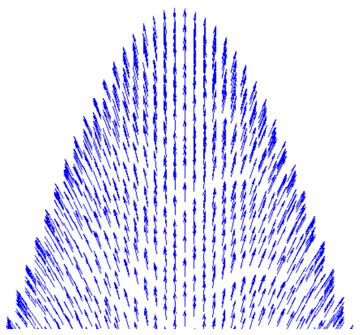

Figure 18: Zoom on the top of the needle map

Now let's plot the images of the three components of the normal vectors.

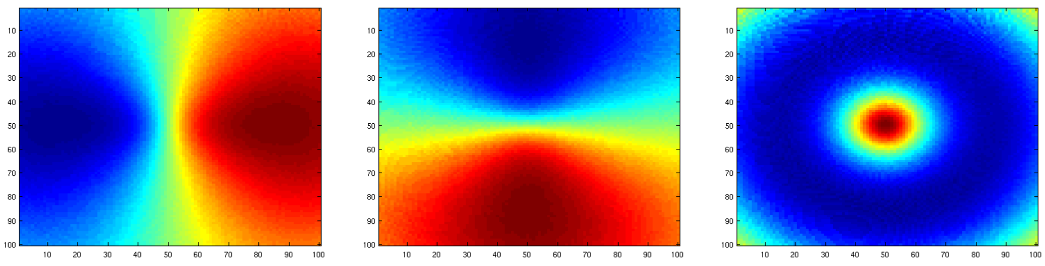

Figure 19: x, y and z components of the normal vector

These images are the raw values of the normal vector components so the sign of the value is just carrying the orientation. So based on the previous images and the patterns for the $x$ and $y$ components that they the shape appears to be symmetrical as we've already noticed with the previous needle map. Here is the absolute value of the images before so we can confirm our analysis.

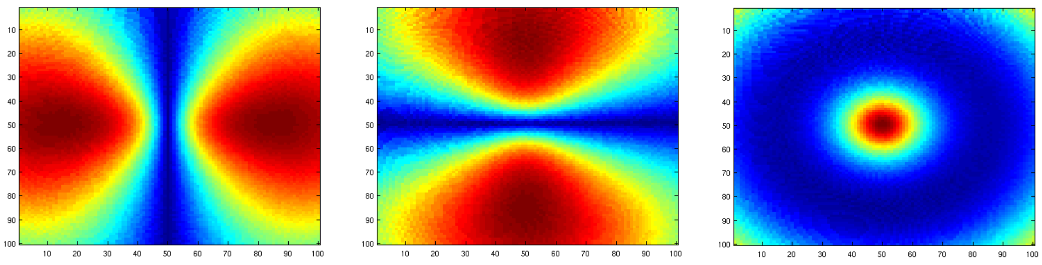

Figure 20: absolute value of x, y and z components of the normal vector

In the previous set of picture we can definitely make sense of the results. In fact if we consider the central point of the image, the $x$ and $y$ components of the normal vector are close to 0 while the $z$ component is maximum, this illustrate the fact that on the top of the shape the normal is directly pointing to the top. We can do the same kind of analysis for any other area of the image.
And finally, we can compute depth and plot the reconstructed surface.

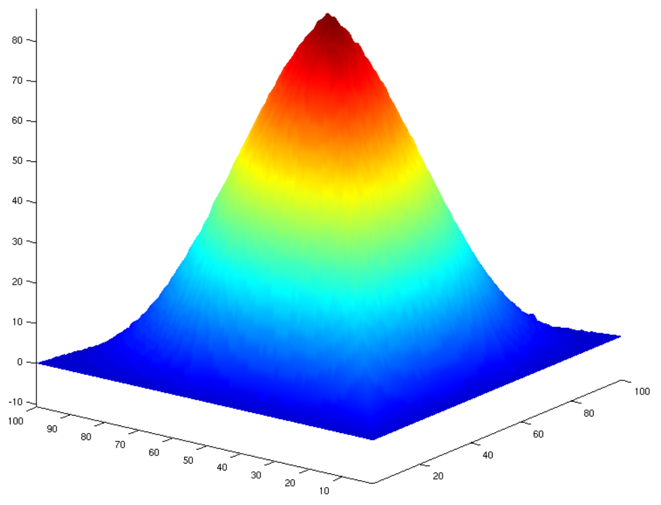

Figure 21: Reconstructed surface based on integration of normal vectors

On the previous image we have displayed the reconstructed surface using normal integration. So in this image we can see the reconstructed shape of the object with a color map according to depth *i.e* the closer is red and goes to blue with depth. (note : color are interpolated). To make sure the normal vectors we computed are normal to the reconstructed surface we can plot them on top of a mesh of the previous surface.

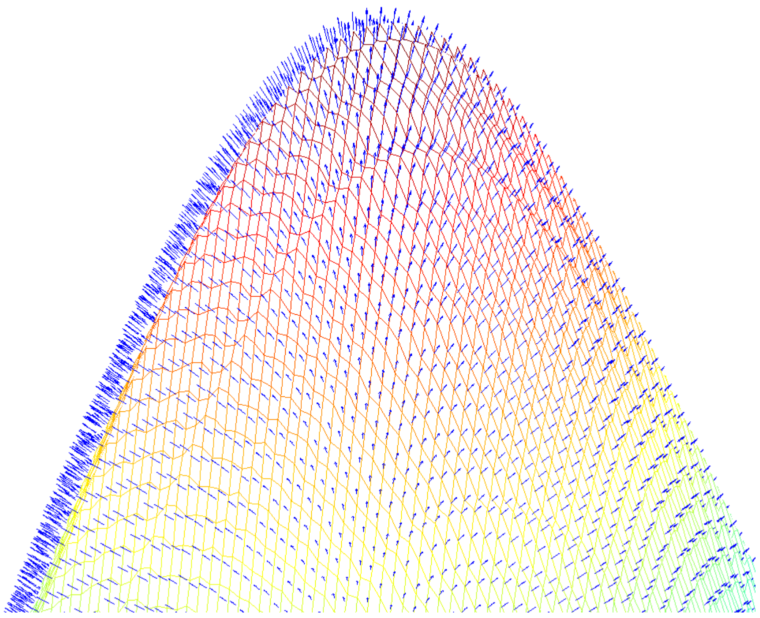

Figure 22: normal vectors on a mesh grid of the reconstructed surface

So finally we can then plot the final surface reconstruction we obtained for this set of images.

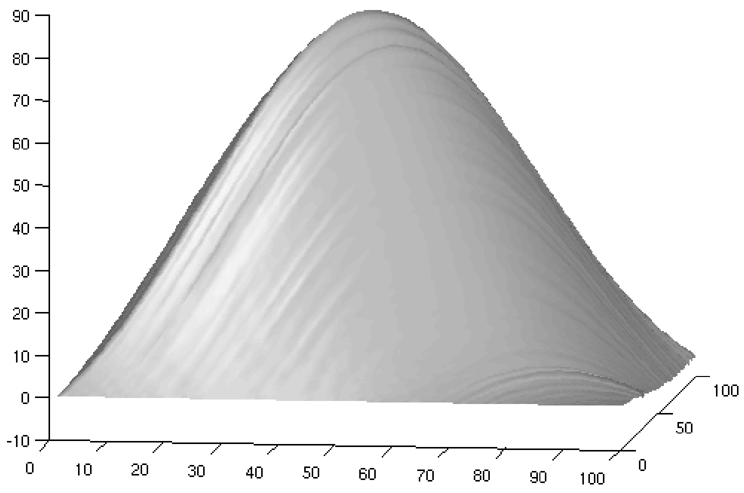

Figure 23: final reconstructed surface

In this reconstruction, we used the four given images, let's quickly take a look at the result if we only used the minimum number of views required *i.e* 3.

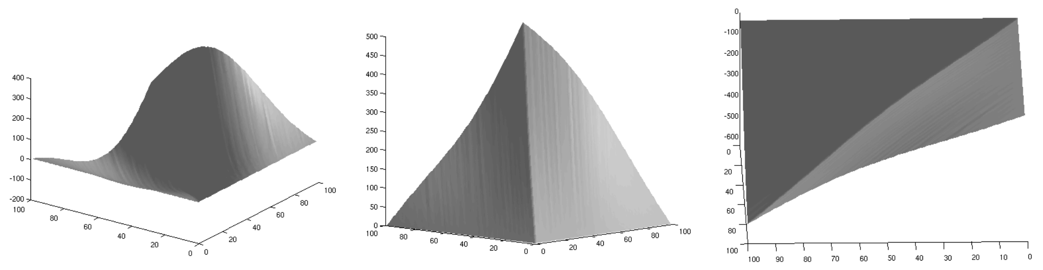

Figure 24: Surface reconstruction without images 1, 2 and 3

The previous image is presenting the results for reconstruction with only three images. The reconstruction without the fourth image is not mathematically possible, the algorithm is failing to compute the surface normals and to reconstruct the surface. Then, if we look at the reconstruction made with only three images, we can see that they are not really accurate. This is probably due to dark areas on several images that prevent the algorithm from reconstructing the normals because our system is not correctly defined.

### 3.1.3 Other data set

We can perform our analysis on an other artificial data set to see how our algorithm is reconstructing the shape. This data set is a sphere so by knowing exactly what shape we should reconstruct we will be able to see the limitations of our algorithm.

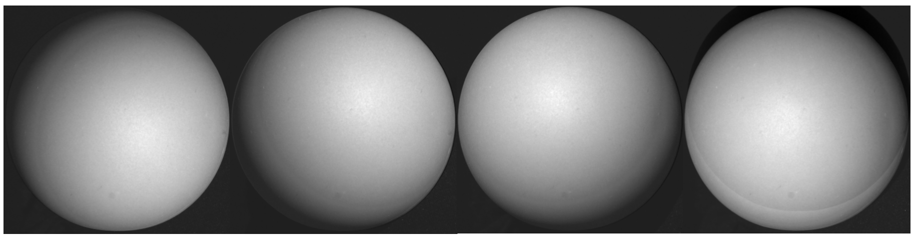

Figure 25: The four images of the second data set

So, here is the estimation of the albedo map associated to this data set:

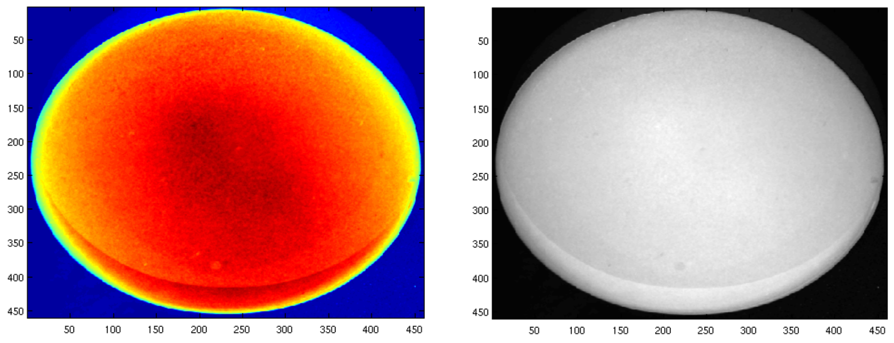

Figure 26: Estimation of the albedo map

The albedo map make some sense regarding to the input images, but the most important thing is the normal vector reconstruction:

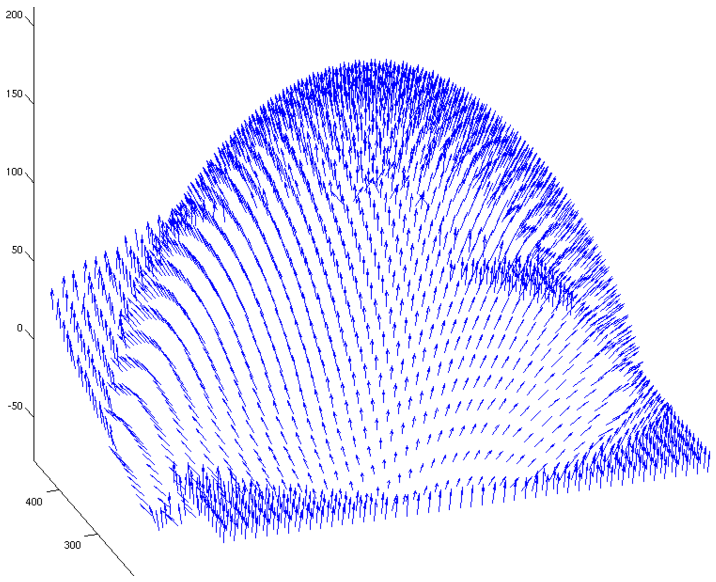

Figure 27: Computation of the normal vectors

As we can see here, the shape appears to be quite good, but some issues also appears with the normal vector estimation. Let's take a look at the shape reconstruction and then we will discuss the issues with the reconstruction algorithm.

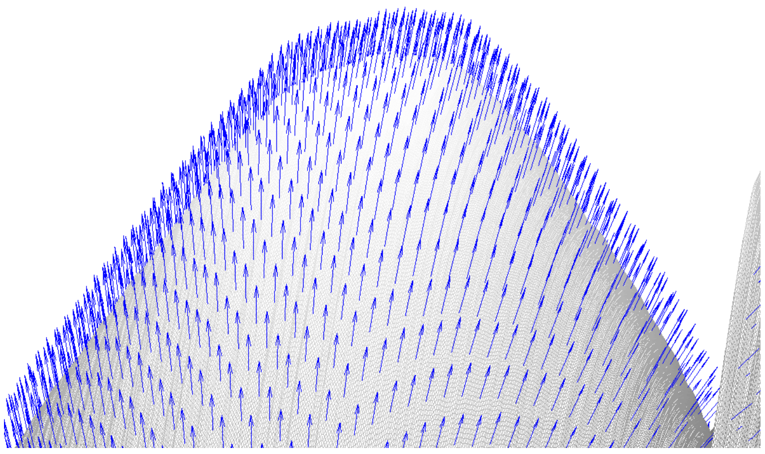

Figure 28: Reconstructed mesh with estimated normal vectors

On the previous picture we can see that on the reconstructed mesh, the normal vectors looks fine. Finally let's present the reconstructed shape:

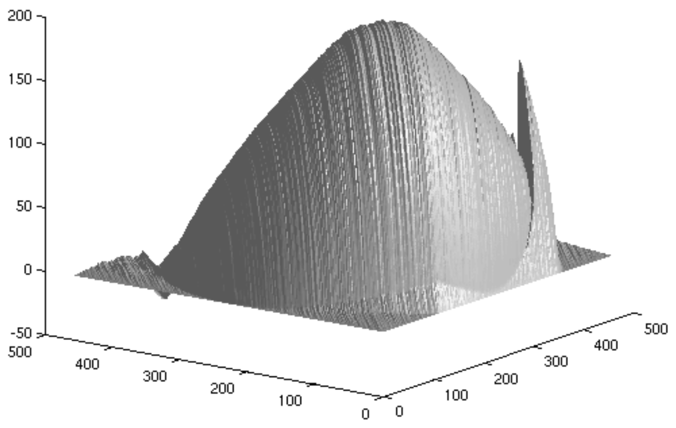

Figure 29: Reconstructed shape of the object

As we can see on the previous picture of the reconstructed shape, the result does not seem to be the best we can obtain. Indeed, there are some "glitches" where the estimated normal vectors are completely wrong which results in the reconstruction of the wrong depth and consequently the wrong

shape. We can clearly see this with the peak on the right of our reconstructed shape. Moreover, the reconstructed shape does not appear perfectly spherical, there seems to be some uncertainty with the obtained shape.
Again we estimated the shape using only 3 images and we also got some issues with the reconstruction. Here are the result by removing one image from the input set.

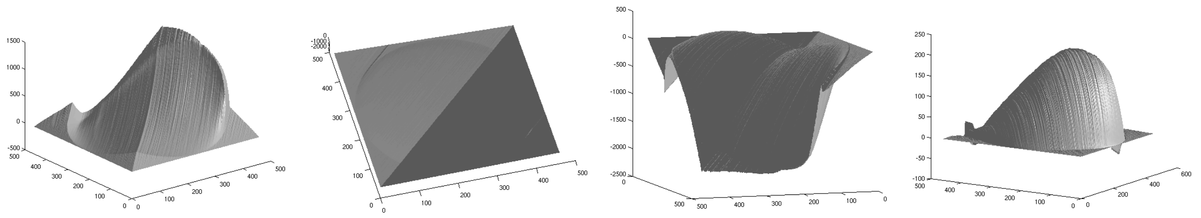

Figure 30: Reconstructed shape of the object without images 1,2,3 and 4

## 3.2 Shape from Shading using real images

The first data set we are going to deal with is a dog data set composed of the following 4 pictures:

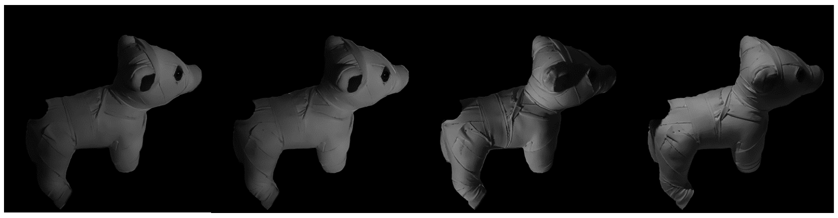

Figure 31: Four dog images with different light source position used to reconstruct shape

Here is firstly the estimated albedo map associated to the set of pictures:

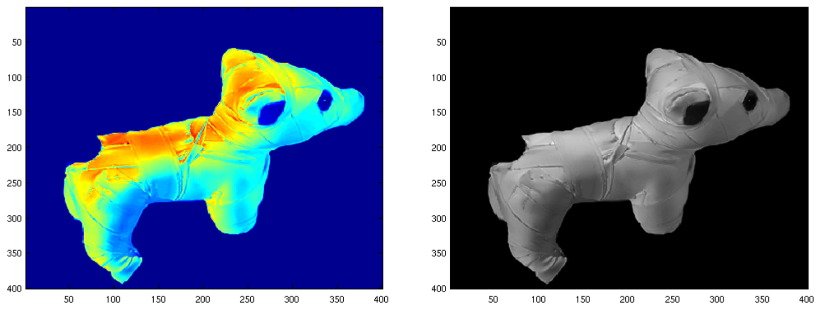

Figure 32: Estimated albedo map

Then we can compute the normal vector associated to the surface:

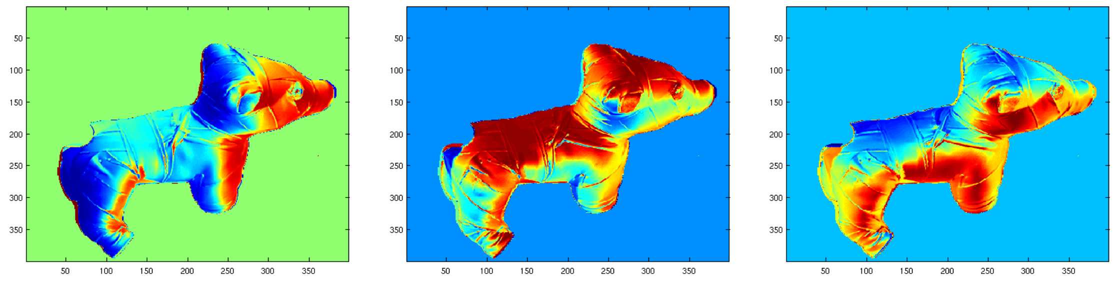
Figure 33: Computed x, y and z components of each local normal vector

Then, we tried to compute the estimated normal vector field and the estimated surface but we ran into serious issues. Here is, hopefully a not too bad result for the vector field:

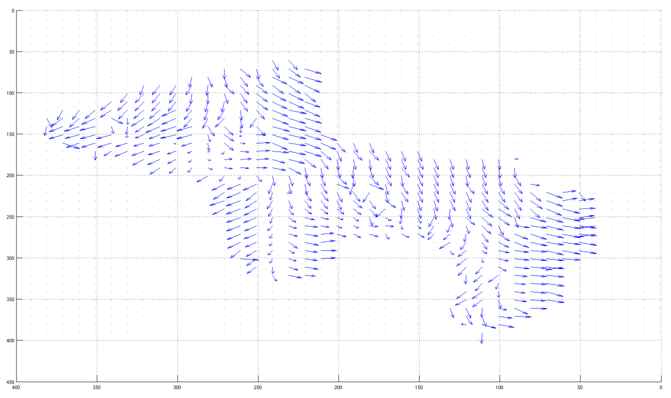
Figure 34: 3D representation of the computed normal vector field

With this data set we can clearly see the limitations of our implementation which fail in reconstructing a correct solution for the shape of the object. We obtain peaks and some pieces of shape but the results are really bad.

### 3.3 Horn method with minimization problem

In this section, we are going to present few results based on Horn method for shape from shading. The objective of this method is to solve the shape reconstruction based on the same set of images we had before. This technique relies on minimizing an energy function. In this case the energy function to minimize is given by:

$$\epsilon = \int\int \Big( (E(x,y) - R(p,q))^2 + \lambda \left( p_x^2 + q_y^2 + p_y^2 + q_x^2 \right) \Big) \, dxdy \tag{43}$$

The resolution of this problem relies on Euler-Lagrange equations :

$$\frac{\partial \epsilon}{\partial p} - \frac{\partial}{\partial x}\frac{\partial \epsilon}{\partial p_x} - \frac{\partial}{\partial y}\frac{\partial \epsilon}{\partial p_y} = 0 \qquad \text{and} \qquad \frac{\partial \epsilon}{\partial q} - \frac{\partial}{\partial x}\frac{\partial \epsilon}{\partial q_x} - \frac{\partial}{\partial y}\frac{\partial \epsilon}{\partial q_y} = 0 \tag{44}$$

We convert this equation system to fit our discrete case of digital images and we implement horn method using Fourier Transform. Please, note that I did NOT fully implement this code myself, I relied on documents provided for this course from Shireen Y. Elhabian (see reference). I wanted to try to provide more results for the shape reconstruction which failed with my implementation for the dog data set. Here is a reconstructed shape for the sphere data set:

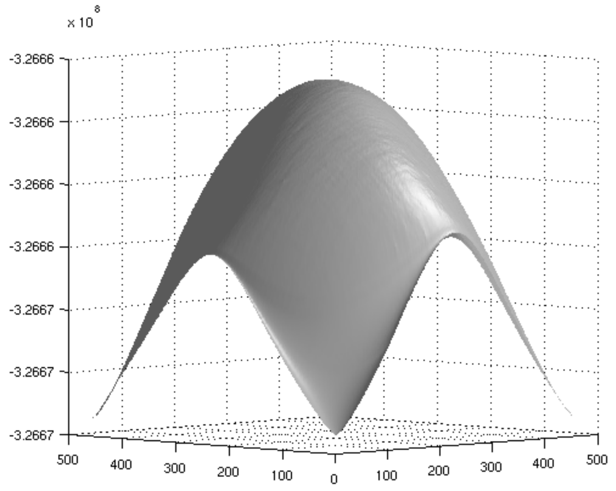

Figure 35: Surface reconstruction using Horn algorithm and Euler Lagrange equations

Note that in this case we did not use the Euler Lagrange optimization algorithm because it seems to diverge so we only did one iteration. The result does not appear too bad. We can clearly see that there is constraint on the smoothness of the shape reconstructed, we can also see a boundary condition acting to prevent the corners of the image to move.

Unfortunately we could not find a way to make it better than the result we presented before. This might come from computational issues.

# 4 Implementation

## 4.1 Shape from Shading using simple integration

Here is our implementation for the shape from shading algorithm.

```
I1=imread('real1.png');
  I2=imread('real2.png');
  I3=imread('real3.png');
  I4=imread('real4.png');

  for i = 1:size(I1,1)
        for j = 1:size(I1,2)
            images(i,j,1) = I1(i,j);
       end
     end

% Initializations

     albedo=zeros(size(I1,1),size(I1,2));
p = zeros(size(I1,1),size(I1,2));
q = zeros(size(I1,1),size(I1,2));
  for i = 1:size(I1,1)
     for j = 1:size(I1,1)
         normal_vector(i,j,1) = 0;
         normal_vector(i,j,2) = 0;
         normal_vector(i,j,3) = 0;
     end
     end

     L=[0,0,1;0.2,0,1;-0.2,0,1;0,0.2,1]; %position of light source
normal=[0;0;0];

% processing
for i = 1:size(I1,1)
        for j = 1:size(I1,2)
            for im = 1:4
            I(im) = double(images(i,j,im));
            end
             A  = L'*L;
             b  = L'*I';
             g  = inv(A)*b;
             albedo(i,j)  = norm(g);
             normal  = g/albedo(i,j);
             normal_vector(i,j,1) = normal(1);
            normal_vector(i,j,2) = normal(2);
            normal_vector(i,j,3) = normal(3);
                p(i,j) = normal(1)/normal(3);
                q(i,j) = normal(2)/normal(3);
            end
    end

    %compute albedo
    maxalbedo = max(max(albedo) );
      if( maxalbedo > 0)
          albedo = albedo/maxalbedo;
      end

      %compute depth
      depth=zeros(size(I1,1));
    for i = 2:size(I1,1)
      for j = 2:size(I1,1)
          depth(i,j) = depth(i-1,j-1)+q(i,j)+p(i,j);
      end
      end

      %display estimated surface
      figure(15);
surfl(depth);
colormap(gray);
grid off;
shading interp
```

## 4.2 Shape from Shading based on Horn Method

Combination of the previous implementation and S. Y. Elhabian, Hands on Shape from Shading, Technical Report, May 2008 code presentation of the method.

## 5 Conclusion

In this third project, we had the opportunity to work and study some important aspects regarding a very important technique which is shape from shading.

This technique, allow us to reconstruct the shape of an object using multiple images where the light source position is changing and according to the object reflecting properties modify the shading. In this project, we presented the theoretical concept used and several results we obtained with our implementation.

We noticed that with the integration technique we used that a bunch of issues occurred. Indeed, as we mentioned, theoretically we only need 3 images to solve our equation system. But most of the time there are dark areas in the image which prevent us from getting complete information about the shape. This resulted in either an unsolvable numerical system due to the lack of information or to results that were completely wrong due to partial information about the shape that were used for computation. We illustrated this aspect by showing how using only three images could affect the reconstruction of the shape of the object.

As we also mentioned, our integration method is pretty simple and is based on normal integration. But this method does not really take in account some properties of the shape as differentiability. Indeed an interesting property of shapes is, in most cases, to be differentiable and continuous. So we need to enforce properties on the second derivative, which is not done in this implementation. So an interesting point of improvement would be to use Horn method based on Fourier transform to enforce this property. Our attempt to do so was not very successful but with more investment we would succeed.

Noise as always with digital images, and processing approximation can be a great source of errors so using multiple images and a Least Square Estimate appears here to be a better solution.

## References

[1] Wikipedia contributors. Wikipedia, The Free Encyclopaedia, 2013. Available at: http://en.wikipedia.org, Accessed March, 2013.

[2] D. A. Forsyth, J. Ponce, Computer Vision: A modern approach, Second Edition, Pearson, 2011.

[3] S. Y. Elhabian, Hands on Shape from Shading, Technical Report, May 2008.

## List of Figures