UNIVERSITY
of
GLASGOW

Computing Science

Extracting Depth Information from Photographs of Faces

Intaek Kim

Submitted to the Department of Computing Science
in fulfillment of the requirements for the degree of

Doctor of Philosophy

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

Recently new methods of recovering the 3D appearance of objects, like stereo-imaging sensors, laser scanners, and range-imaging sensors provide automatic tools for obtaining the 3D appearance of an object but they require the presence of the object. When only photographic images are available, it is still possible to reconstruct the 3D appearance of the object if there is also a model which can be referenced.

The human face is very popular with researchers who try to solve the problems including facial recognition, animation, composition, or modelling. However it is rare to find attempts to reconstruct shape from single photographic images of human faces, although there are numerous methods to solve the shape-from-shading (*SFS*) problem to date.

This thesis describes a novel geometrical approach to reconstructing the original face from a very impoverished facial model[1] and a single *Lambertian* image. This thesis also introduces a different approach to the *SFS* problem in the sense that it uses prior knowledge of the object, the so-called *shape-from-prior-knowledge* approach, and addresses the question of what degree of impoverishment is sufficient to compromise the reconstruction.

Most recovered surfaces using conventional *SFS* methods suffer from flattening so that we cannot view them in other directions. We believe that this flatness is due to the lack of geometric knowledge of the subject to be recovered. In this thesis, it is also argued that our approach improves upon existing *SFS* techniques, because a reconstructed face looks correct even when it is turned to a different orientation from the one in the input image.

[1] This is a polygonal mesh that poorly represents the geometrical structure of the original face.

Contents

List of Figures

List of Tables

Acknowledgements

First I would like to thank the Republic Of Korea Air Force for allowing and supporting me to complete this study for almost five years in Glasgow, United Kingdom. Especially during the last year, when in spite of the state economy being in a difficult situation, an extension period to complete this work was granted. I am indebted to all those who supported me at the Division of Education and Training and at Air Force Academy.

I would like to thank my thesis supervisors Dr. John W. Patterson and Phillip D. Gray, for their excellent advice and constant encouragement through my doctoral research. They have not only taught me computer science but have also given me a lifetime philosophy to pursue for working on real world problems. Especially meeting John every week was an uplifting experience for me - both intellectually and spiritually.

In preparing this thesis, special thanks from the bottom of my heart are due to Prof. Christopher W. Johnson and Dr. Jean-Christophe Nebel. Their systematic, devoted, and well-organized comments on the early drafts of this thesis have been and shall remain a major source of inspiration to me. I would like to express sincere appreciation to James E. M^cGhee who provided many useful comments on the thesis. Also I wish to thank indeed the examining committee members, Dr. Adrian L. Thomas from the University of Sussex and Dr. Paul Siebert from the Turing Institute and the viva panel convenor, Prof. Simon L Peyton Jones.

My wife Jeunglim and daughter Yangheun patiently put up with several years of fatherless nights while I was preoccupied with *Lilybank Gardens.* My duty to Yangheun in her school life was always shifted on to my wife. Without her help, neither this thesis nor I would be what I am. I should have been home with my family for Chrismas and the new year's eve. Now I can devote full time to the infinitely more important issues of family relationships. I also would like to thank my parents-in-law, brother, and sister for their encouragement and devotions during our long absence.

I thank God for all the blessing He has bestowed upon my family. Finally, the thesis is dedicated to the memory of my parents.

Chapter 1

Introduction

In this thesis we present a new alternative approach concerning the shape-from-shading (*SFS*) mechanism, which may give rise to some answers. We believe that human brain uses internal models, based on expectation or prior knowledge [Yin69, Ram88a, Ram88b, BET94, Val95, Spe96], in order to understand three-dimensional (3D) shape from two-dimensional (2D) shading. We give a concrete set of examples employing internal models of 3D objects, based on human faces.

1 Problem Statement

The technical way to obtain 3D facial information is to take manual measurements of selected points which have been marked on the face we want to analyse, or on the surface of a plaster model of a face. This is a time consuming activity that is tedious if the measurements of a number of faces are required.

Recently, the development in the stereo-imaging sensor [SU94] provides an automatic tool for obtaining 3D face models. The technique for obtaining them including the laser scanner [DY88, Lin93] or the range-imaging sensor[TK92] requires that we be presented with a subject in order to obtain 3D data.

However a worse case - a some different case - is when the subject for which we want measurements is not present and only photographs are available, for example in the case of a fire victim's face, a dead person's face, or a missing child's face. In these cases we cannot retrieve the 3D face information from the previous techniques. This thesis concentrates its efforts on a new method for handling these difficult cases by bringing together shape from shading techniques with the kind of prior knowledge that appears to be used in human visual perception.

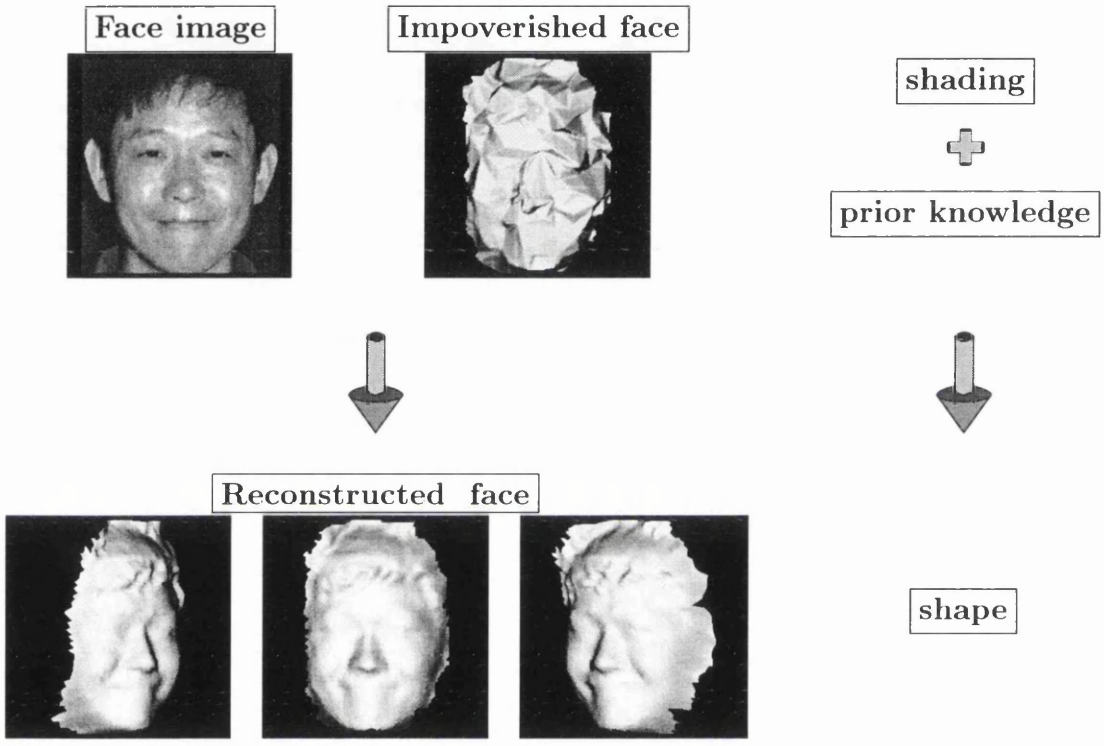


Figure 1.1: An ideal example of the face reconstruction

The goal of this thesis is to rehabilitate an impoverished face, upper right, of a person, in conjunction with a single face image, upper left. The reconstructed face, bottom, approximates to the original face portrayed in the face image. It should be right even when it is turned to a different orientation as the one in the face image.

Let us try to understand more technically the goal of this thesis; it can be stated formally as follows:

- Given a single image and a coarse model of a human face, reconstruct an approximation of the original face of the image.

Figure 1.1 shows an example of the problem domain to be solved in the progress of our investigation. The upper-left picture is a single face image illuminated by a light source. The direction of the light source is known in advance. The upper-right picture is a representation of an impoverished face as the prior knowledge.

As you see in the figure, the impoverished face is no longer considered as the original face of the face image, but it only preserves the global topology of the original face. By using information about the impoverished face, the shape-from-shading (*SFS*) problem changes from being ill-posed to well-posed. Starting from this impoverished face, the original face portrayed in the face image is inversely reconstructed in the form of an approximation. The bottom pictures are the target; that is, a reconstructed face model that can be used to generate different views.

In conventional approaches to the *SFS* problem, it should be stressed that the recovered surfaces are unlikely to support different views, because they are too flat[1] like the bas-relief images on coins. That is, most attempts to solve the *SFS* problem concentrate on the recovery of images with different light source directions. However the reconstructed faces in this thesis can be seen in different views.

2 Background

Our perception of the 3D shape of a surface depends on many cues including shading, perspective, texture, stereopsis, and motion parallax [FS94]. Among them, it is well known that the human visual system is proficient in perceiving the 3D shape of a surface from shading in a 2D image [Ram88a, Ram88b].

How the *SFS* mechanism is carried out is a more interesting problem. There have been a number of research efforts to realise the mechanism on computers over last several decades, yet it is not well understood from a standpoint of computer applications.

[1] Figure 2.11 in *Chapter 3* shows an example of the flatness.

2.1 Deep-seated problems in shape from shading

Shading is the smooth variation in intensity from one point to another in an image. We could get 3D shape information from this shading if we make some assumptions about the imaging model or surface property.

For example, the amount of light reflected by a surface patch depends upon only its normal vector, the so-called *Lambertian* surface[2]. In practice, however an intensity value alone cannot yield a unique solution to the surface normal at a given image point, even though we employ the simplest case of imaging model such as the *Lambertian* surface. Since there are an infinite number of solutions that correspond with the intensity value, the problem is called ill-posed[3].

Therefore, we must find additional constraints on the imaging model in order to solve the ill-posed problem. Smoothness and integrability are frequently used as the additional constraints. For example, variational approaches attempt to minimise the error with these constraints [IH81, BH85, FC88, Hor90, Sze91, VY93]. Other assumptions about local surface shape are also used as the additional constraints in local approaches, which attempt to find an approximate solution using the pre-assumed shape in the vicinity of an image point [Pen86, LR85].

All these approaches imply that the surface orientations of neighbouring surface patches are strongly correlated, and that the surface is continuous and smooth. Most methods proposed up to this time have made the *SFS* problem well-posed by enforcing these additional constraints so that they may cut the infinite number of solutions corresponding to an image intensity[4] down to the few that satisfy the constraints. We will review the methods for the *SFS* problem proposed so far in detail in *Chapter 2.*

The *SFS* methods proposed so far in the literature suffer from problems including multiple false solutions, the sensitivity to image noise, the flatness of reconstructed surfaces, the use of complex mathematics, and large numbers of iterations, even when applied to the *Lambertian* surface which is the simplest imaging model.

[2]For example, the reconstructed faces in Figure 1.1 are images of *Lambertian* surfaces. The imaging model about this surface is addressed in *Chapter 2* more in detail.

[3]Conventionally the heart of the *SFS* problem is the solution of the imaging model or image irradiance equation. It is $I(x,y) = R(p(x,y),q(x,y))$, where $I(x,y)$ is an observed intensity of an image point, $R(p,q)$ is the reflectance map (scene radiance) for a surface point, and p and q are partial derivatives of the surface height with respect to x and y. Therefore, we must find additional constraints on the imaging model in order to solve that ill-posed equation, since it contains two unknowns p and q at a particular point in the image.

[4]The term `image intensity` will be used in this thesis rather than the technically more correct `image irradiance`.

These limitations are inevitable in the sense that the *SFS* is an ill-posed problem and hence require additional assumptions. In this thesis, among those, we try to overcome the limitation of the flatness by applying our method so that the reconstructed faces can be viewed from different directions.

2.2 Internal models? : As prior knowledge

One of the most interesting abilities of our vision system is that of face understanding. For example, the original face in a passport photograph can typically be recognised by most people looking at it. As another example, a sculptor is able to reconstruct the original face in wood or stone from a photograph, the so-called portrait sculpture [Maz94]. It seems probable that they use internal models of faces in doing so [Par91, BB93, Val95, Spe96].

2.2.1 A feature-based model

An upside-down face is recognised more poorly than an upright face [Yin69, BET94]. Figure 1.2 shows an example of the upright and the upside-down face image. We can easily perceive the upright face on the left-hand side, however it takes longer to do the upside-down face in the middle. This difficulty reveals that we are accustomed to an upright face orientation.

Another inversion effects were demonstrated by Thompson [Tho80, Ste95]. In an upright face of Margaret Thatcher's photograph, he inverted the mouth and the eyes. The resultant face as shown on the right-hand side in Figure 1.2 is not perceived properly due to the incorrect feature orientation.

These demonstrations suggest that our vision has a feature-based internal model, which is constructed through interaction with our own prior experience or expectation, in interpreting visual data. There are many approaches to automating face recognition based on this feature-based model [CEL87, MCvdM92, BP93].

2.2.2 A geometry-based model

Figure 1.3 shows another interesting example of the use of prior knowledge, assuming that there is a distant light source in the right direction for the images[5].

[5]These images are created by applying the normal interpolation renderer to polygonal faces, which is addressed in *Chapter 2.*

Figure 1.2: Inversion effect on face recognition: An example

The left-hand side shows an upright facial image and the others show inversion effect. We can easily perceive the face on the left, however we have difficulty in doing the upside-down face image, middle, and the eyes-and-mouth-rotated one, right. The difficulty indicates that we are accustomed to the upright orientation of a face and its features. This suggests that our vision has a feature-based internal model, which is constructed through interaction with our own prior experience or expectation, in interpreting visual data.

The face on the left-hand side is a convex-shape, while the right face is a concave-shape. The convex-shape means that the geometry of the face is normal as we know it. For example, the nose is nearer than the eyes, and so on. On the other hand, the concave-shape is negatively equal to the convex-shape. Namely, the nose of the concave-shape face is dented, which is farther away than the eyes, and so on.

As seen in Figure 1.3, the right image could be interpreted as the concave face rather than as the convex one. That is to say, we interpret the light illuminating from the opposite direction to that in the case of the convex face. This `prior knowledge` could be regarded as a geometry-based internal model used in the portrait sculpture.

This idea allows us to introduce an alternative approach in order to overcome the deep-seated problems in the *SFS*. The idea of our new method is to use geometric-impoverished-models, for example the impoverished face shown in Figure 1.1, to represent prior knowledge about the structure of faces. Now the *SFS* can be reformulated as the shape-from-prior-knowledge problem to be presented in this thesis.

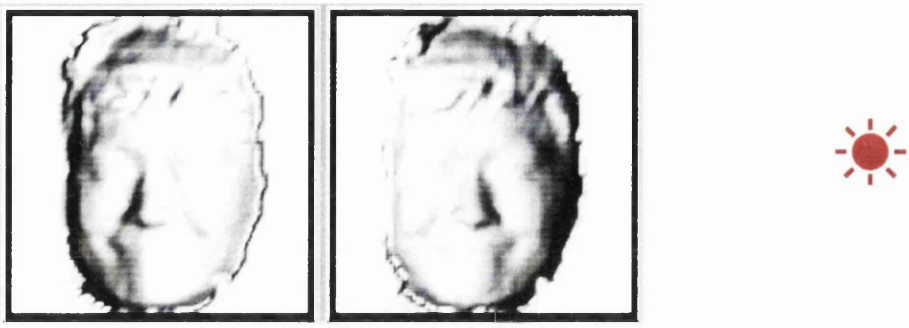


Figure 1.3: Convex and concave facial image: An example

For a given point light source directed in a small right angle, the face images are synthesised by a convex-shape mesh (on the left) and a concave-shape mesh (on the right). Our perception is reluctant to interpret the right face as the concave-shape face and is willing to interpret the light as being in the opposite direction instead.

3 Limitations

Two factors are critical for solving the inverse shading problem in general: the imaging model and the light source. The imaging model specifies how an image intensity is formed by the dependence on the surface geometry under given circumstances. Previous research has mostly been focused on the property of diffuse and specular reflection with the assumption of orthographic projection [HB89, Nal93, CSO97, WH97].

The imaging model we use through this thesis employs the diffuse reflectance property, also known as the *Lambertian* surface, with orthographic projection. The *Lambertian* surface reflects the light diffusely and an intensity at a particular point is propotional to the cosine of the angle between the surface normal and the incident light direction. This imaging model turned out to give a fairly realistic approximation for human faces [IS91]. We will discuss this topic in *Chapter 2* more in detail.

The light source is an object emitting radiant energy which contributes to the variation of intensities on an image. It is a widely accepted assumption that there is only one distant-light-source illuminating a fairly large image [Ram88b]. Under this assumption, researchers have developed many algorithms determining the light source direction [Pen82, BH85, LR85, ZC91]. Doing so is not a major

problem in this investigation, because we investigate the reconstruction of the original faces from shading and impoverished geometric faces.

In order to simplify our task, we assume that the light source direction will be given in advance. In addition, we will assume that the reflectance coefficient is constant in the original faces. This constant assumption has turned out to be a good approximation [AGR95].

Our approach also relies upon a 3D surface representation. Although there are many methods to represent the 3D surface of objects, we employ a smoothly interpolated polygonal mesh formed from triangles. The geometrically impoverished face as the prior knowledge is, of course, represented by this polygonal representation. An impoverished face is obtained by our own mesh simplification method, which is fully discussed in *Chapter 3.*

In summary, the shape-from-prior-knowledge problem will be discussed in terms of attempting to reconstruct the original faces fused in single photographs or *Lambertian* images, starting from an impoverished face. The images do not have specular components, although such components may be well represented in shading formulae in computer graphics. Furthermore, we no longer employ the additional assumptions on the smoothness, the integrability, or local shape.

4 Originality and Main Contributions

This thesis has addressed the problem of reconstructing a human face, starting from the level of impoverished face. This impoverished face is revised in the iterating process so as to reflect a polygonal approximation of the original face in a face image.

In this thesis, the principal contributions achieved in solving the problem are as follows:

- ❒ It proposes a new geometrical approach to reconstructing an approximation of the original face from single images of human faces.
- ❒ It introduces a new alternative to shape-from-shading in the sense that it uses the prior knowledge of the geometry of subjects such as an impoverished face, called *shape-from-prior-knowledge.*
- ❒ It presents a new tool for mesh simplification in order to simulate impoverished levels of a human face, called the *Meducer.*
- ❒ It presents a new tool for editing a polygonal mesh, called the *Meditor.*

- It employs a quaternion method in the process of reconstructing a face.

5 Overview of Thesis

The main organisation of this thesis is as follows: in *Chapter 2* we will review the issues related to our investigation, which have been proposed so far in the literature. They are the shading model[6], the *SFS* methods, and finally face reconstruction methods.

The first issue is related to the face images to be reconstructed. We use synthesised images rather than real images. That is, a face image is created by the *Lambertian* shading model. The *SFS* methods can be classified into three groups. The main contributors in the groups are implemented and addressed with an image example. The techniques for reconstructing human faces, which are directly measurable, are then reviewed.

The main contributions start from *Chapter 3*. The discussion in *Chapter 3* concerns impoverishing faces. It includes two topics: *Meducer* and *Meditor*. These are tools to provide a level of an impoverished face. The *Meducer* is for impoverishing original faces in terms of their triangles as much as some level we need. On the other hand, the *Meditor* provides an interactive way of correcting the undesired triangles created in impoverishing a face using the *Meducer*.

In *Chapter 4*, we reformulate the conventional *SFS* problem into the shape-from-prior-knowledge one. The surface normals corresponding to each intensity in an image are calculated in a totally different way comparing with the *SFS* methods.

In *Chapter 5*, we discuss in detail the face reconstruction procedure, using the surface normals obtained in *Chapter 4*. The discussion starts with defining two quaternions and introduces a quaternion method in the process of reconstructing a face. The procedure is explained in a step-by-step fashion and demonstrated for a very impoverished face.

In *Chapter 6* we define statistical estimators. Using them the face reconstruction method is analysed with a number of examples including differently impoverished levels of a face, different person's faces, and a half cylinder and flat surface as impoverished faces.

[6]This term is usually referred as the imaging model or the rendering model. The former is focused on computer vision, which is also called the reflectance model, while the latter is focused on computer graphics. In this thesis, we prefer to use the shading model for both sides.

Finally in *Chapter 7* we give conclusions and examine possible directions for future work. Appendices also provide additional explanations for some chapters.

Chapter 2

Related Work

As we mentioned in the previous chapter, the aim of this thesis is to develop methods for reconstructing a model of the original face from a single shaded image of a human face, starting from an impoverished face in the form of a polygonal mesh.

The related work for the achievement of our aims includes issues concerning the following:

- shading (imaging, reflectance) models,
- inverse shading (*SFS*) methods,
- face reconstruction methods, and
- mesh representation and impoverishment methods.

The final issue is presented separately in the next chapter and the rest are reviewed in this chapter.

1 Shading Models

Shading is a continuous variation of image intensity, which appears to be 3D [Ram88a] because it is determined by the surface geometry and the reflectance property of an object. Given surface geometry and reflectance property, computer graphics simulates the shading in order to synthesize realistic images [Bou70, Gou71, Pho75, Whi80, CT82, CG85, Kaj85, HTSG91, ON94].

Viewed the other way, given real or synthesized images, computer vision and image analysis techniques recover the surface geometry and the reflectance property [HB88, HB89, Hor90, NIK90, ZC91, LK93].

Most existing techniques in both approaches only work for simple surfaces; for example, *Lambertian*, plastic, and metallic ones. It is one reason that shading in the real world is too complex to understand completely. This complexity has resulted in a great number of the shading models in the computer literature [BS63, TS67, CT82, KvD83, FZ91, HTSG91, NIK91, Wol92, ON94]. At this point, it is worth mentioning that there are excellent surveys of the shading models [Hor81, Hal86, Lew94, Sch94].

Lots of the shading models that have been proposed so far describe the behaviour of light at a surface point. Their arguments are mostly to identify three components concerning light reflectance as shown in Figure 2.1:

- diffuse,
- specular lobe, and
- specular spike reflectance[1].

With these components, an image intensity $I_{\vec{V}}$ can be computed in terms of the sum of their contributions on a surface point in a viewing direction $\vec{V}$ [HTSG91, NIK91] as follows:

$$I_{\vec{V}} = I_d + I_{sl} + I_{ss},$$

where I_d is a diffuse reflectance component, I_{sl} is a specular lobe reflectance one, and I_{ss} is also a specular spike one.

Here we will geometrically and separately review these components, which were derived from a number of the existing shading models, in order to develop a basic understanding of how an image intensity is computed. In this respect, we will also make assumptions related to a distant point light source and a distant viewer.

1.1 Diffuse reflectance

Diffuse reflectance occurs if light is reflected in all directions on a surface point. In other words, surfaces appear equally bright in all directions. As shown in Figure 2.1, its appearance is due to internal scattering.

[1]The lobe and the spike component are collectively called specular reflectance.

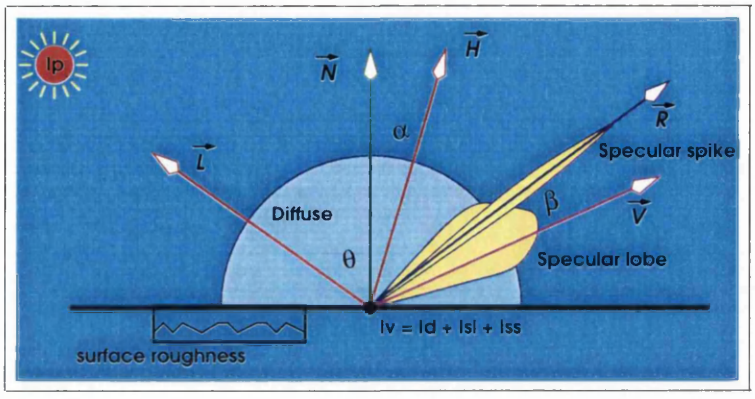


Figure 2.1: Reflectance components

For a given point on a surface, an image intensity $I_{\vec{V}}$ can be generally described with three reflectance components: diffuse, specular lobe, and specular spike. The diffuse component appears round due to the equally reflected light in all directions, and the specular components appear sharply due to the directionally reflected light. The image intensity $I_{\vec{V}}$ in a viewing direction $\vec{V}$ is the sum of them; $I_d + I_{sl} + I_{ss}$.

Incident light[2] hits a surface and encounters microscopic irregularities in the surface medium and is then internally scattered at boundaries between regions of differing refraction. Some of the scattered light rays find their ways to the vicinity of the surface at random [HS89]. When this internal scattering produces a constant image intensity in all directions, it is called perfectly diffuse reflectance or a *Lambertian* reflectance model.

On a given surface point, an image intensity due to the *Lambertian* reflection I_d, as referred in Figure 2.1, is proportional to a cosine of an angle θ between a surface normal $\vec{N}$ and a light source direction $\vec{L}$ as follows:

$$\begin{aligned} I_d &= I_p k_d \cos\theta \\ &= I_p k_d \frac{\vec{N} \bullet \vec{L}}{|\vec{N}| \cdot |\vec{L}|} \\ &= I_p k_d (\vec{N} \bullet \vec{L}), \end{aligned} \tag{2.1}$$

where I_p is the intensity of a light source, k_d is a reflectance coefficient of the surface, and the magnitude of a vector $|...|$ is equal to one.

[2]This is a collection of light rays from a source to a surface point [Gla89]. In the thesis, we will refer it as a derectional vector $\vec{L}$ from a surface point to the source instead, as shown in Figure 2.1.

The shading model considering only this component has been used extensively to develop the *SFS* methods [HB89, Hor90, Pen90, LB91, ZC91, OD93, LK93, Lan94, AGR95] and the photometric stereo methods [Woo77, Woo80, Woo81]. They can produce a smooth shape surface recovered from *Lambertian* images.

This diffuse reflectance component alone turns out to be a fairly realistic approximation for many surfaces including human faces. For example, *Ikeuchi* and *Sato* [IS91] have quantitatively confirmed that human faces are dominated by this diffuse reflectance.

1.2 Specular reflectance

Specular reflectance is made due to unequally reflected light on a surface point. While the diffuse reflectance does not depend on the location of the viewer, this does depend on it. Again, more light is reflected in some directions is than that in others.

As shown in Figure 2.1, incident light $\vec{L}$ is mainly reflected in the direction of $\vec{R}$ which is $\vec{L}$ mirrored by $\vec{N}$. This specular reflectance component, so called I_s, contributes to a highlight on a surface. It is again distinguishable into a specular lobe component I_{sl} and a specular spike component I_{ss}.

The specular lobe component I_{sl} is an amount of light reflected and distributed around the mirrored direction $\vec{R}$, which is dominant on a rough surface. It can be observed differently depending on a viewing direction $\vec{V}$.

On the other hand, the specular spike component I_{ss} is an amount of light perfectly reflected in a very small region around the direction of $\vec{R}$, which is dominant in the case of a metallic-smooth surface and facial skin. It can be seen only in the direction of $\vec{R}$.

In the next subsections, two families of shading model are reviewed with an attention to the specular reflectance component: empirical models and physical models.

1.2.1 Empirical models

Phong introduces an ad hoc empirical model in order to deal with specular reflectance in computer graphics [Pho75], so-call *Phong* shading model. In his model, the specular lobe and spike are expressed as a single highlight function

without discrimination. The highlight function I_s is specified by the powers of the cosine of an angle β between $\vec{R}$ and $\vec{V}$ as follows:

$$\begin{aligned} I_s &= I_p k_s \cos^n \beta \\ &= I_p k_s \left(\frac{\vec{R} \bullet \vec{V}}{|\vec{R}| \cdot |\vec{V}|} \right)^n \\ &= I_p k_s (\vec{R} \bullet \vec{V})^n, \end{aligned}$$

where I_p is again the light intensity, k_s is a specular coefficient, the magnitude of a vector $|...|$ is equal to one, and n is a measure of surface roughness varing from 1 to infinite.

This highlight function is maximum when the angle β is zero, and falls off sharply as β increases. In other words, a specular reflectance in a viewing direction $\vec{V}$ depends on the angle β.

On the other hand, a small value of n represents a broad specular reflectance for a rough surface, whereas a larger value represents a narrower specular reflectance for a smoother surface. The former can be regarded as a specular lobe component and the latter can be considered as a specular spike. In an extreme case, a specular spike appears when n is infinite.

Figure 2.2, which was adapted from [HB94], shows some examples of the sharpness of $\cos^n \beta$ according to values of n. These can be observed in the highlight function. When n is equal to 2 or 8, it simulates a specular lobe. In addition, when n is equal to 64 or 256, it simulates a specular spike.

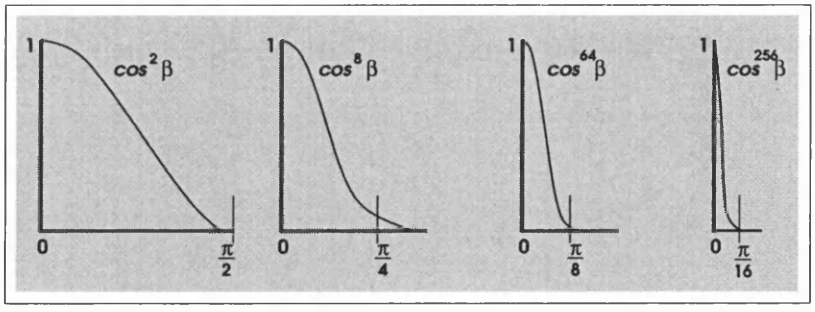


Figure 2.2: Sharpness simulation of specular reflectance in *Phong* shading model [HB94]

The first two from the left-hand side can be regarded as a specular lobe component and the rest can be considered as a specular spike.

Instead of $\vec{R}$ in the *Phong* shading model, *Blinn* used a halfway vector $\vec{H}$ of the sum of $\vec{V}$ and $\vec{L}$, called *Blinn* shading model [Bli77], as shown in Figure 2.1. So the highlight function I_s is rewritten as:

$$\begin{aligned} I_s &= I_p k_s (\vec{N} \bullet \vec{H})^n \\ &= I_p k_s (|\vec{N}| \cdot |\vec{H}| \cos\alpha)^n \\ &= I_p k_s \cos^n \alpha, \end{aligned}$$

where $\vec{H} = \frac{\vec{L}+\vec{V}}{|\vec{L}+\vec{V}|}$, and $\vec{N}$ and $\vec{H}$ are normalised. The angle α between $\vec{N}$ and $\vec{H}$ is similar to β. The halfway vector $\vec{H}$ provides a faster method in rendering an object, because $\vec{L}$ and $\vec{V}$ are fixed, while $\vec{R}$ varies depending on $\vec{N}$.

These empirical models collectively imply a specular lobe and a specular spike component in their highlight functions. They are widely used to render realistic images of various surfaces in computer graphics, because they provide a specular lobe for rough surfaces and a specular spike for smooth ones.

In addition, they are usually efficient in computation without any exact value of a light source intensity. Several areas for special effects in movies, video art, or commercials, in which such quantitative value is not required, are suitable for these models[3]. However, they often produce cartoon-like images, since they have no physical model for surface roughness and detailed light properties. Physical models refine this problem by employing a microfaceted model of surface roughness and by employing a microwave theory of light properties [CT82, HTSG91, NIK91], which are described in the next subsection.

1.2.2 Physical models

These models are based on the physical property of surface roughness and light. This thesis focuses on the former approach because it has the greatest impact upon facial reconstruction. Readers who are most interested in light models should refer to [BS63, NIK90, NIK91].

There are several ways in modeling the surface roughness as distribution functions of primitive shapes. Spherical and cylindrical cavities were first used to derive a shading model on the lunar surface [Hap66, BWR68]. V-groove cavities were also used to deal with a shading model describing metallic surfaces [TS67]. Moreover, hairy cylinders were used to model anisotropic surfaces [PF90].

[3]For example, *Parke* and *Waters* used the *Phong* model to render their face models [Par82, Wat87].

Here we employ a *Cook-Torrance* shading model in order to describe the specular reflectance in physical models [CT82]. This approach employs the *Torrance-Sparrow* surface model. Each surface is composed of small symmetric V-grooves, so called microfacets, each of which perfectly reflects light which falls onto it [TS67].

Figure 2.3 shows a roughness profile of a surface together with several vectors to be used in driving the specular component. $\vec{n_f}$ is a normal vector of a microfacet. $\vec{N}$ is a mean normal vector of $\vec{n_f}$'s. The dashed line represents an imaginary mean surface perpendicular to the vector $\vec{N}$. $\vec{L}$ is a directional vector pointing to a light source, $\vec{V}$ points to a viewer, and $\vec{H}$ is a halfway vector between $\vec{L}$ and $\vec{V}$. In addition, α is an angle between $\vec{N}$ and $\vec{H}$, and δ is between $\vec{H}$ and $\vec{V}$. We assume that the magnitude of those vectors is equal to one through this section.

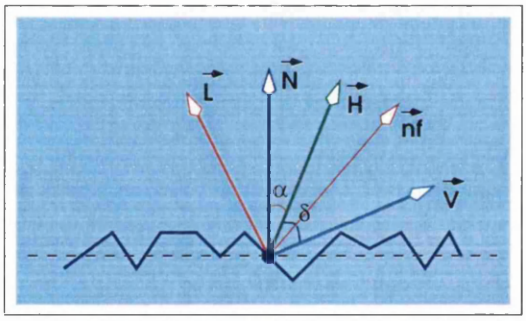


Figure 2.3: Surface roughness

A large set of microfacets constitutes a surface. The surface has a mean normal vector $\vec{N}$ and each microfacet has its own normal vector $\vec{n_f}$.

Similar to using a cosine function for the specular reflectance in the empirical models, *Cook* and *Torrance* proposed a specular reflectance I_s that is fused by a specular lobe together with a specular spike as follows:

$$I_s = I_p k_s \frac{FDG}{(\vec{N} \bullet \vec{L})(\vec{N} \bullet \vec{V})}, \tag{2.2}$$

where F is a *Fresnel* term, G is a geometrical attenuation factor, D is a slope distribution function, and the rest of the coefficients are used as usual.

1.2.2.1 F: Fresnel term The *Fresnel* term F describes a colour change of the specular reflectance. This is an interaction function between a material and an incident light. It depends both on a light wavelength λ and on an incidence angle δ. This colour shift can be observed on a perfectly smooth and mirror-like surface.

Cook and Torrance adopted a *Fresnel* equation to directly obtain an angular dependence of F for a fixed-wavelength λ as follows:

$$F = \frac{1}{2}\frac{(g-c)^2}{(g+c)^2}\left(1+\frac{(c(g+c)-1)^2}{(c(g-c)+1)^2}\right),$$

where $c = \cos\delta = \vec{V} \bullet \vec{H} = \vec{L} \bullet \vec{H}$ and $g^2 = n^2 + c^2 - 1$, in which n is an index of refraction of a surface for a wavelength λ. We assume that F varies according to a viewing angle δ, because c and g are also dependent on δ. This indicates the angular dependence.

In general n is not available. In this case, it can be obtained from F_0, which is a *Fresnel* term measured when $\delta = 0$; that is, $\vec{L} = \vec{V}$. Subsequently $c = \vec{V} \bullet \vec{H} = \vec{L} \bullet \vec{H} = 1$. Therefore $g = n$ $(g, n > 0)$, and hence F_0 satisfies as follows:

$$F_0 = \left(\frac{n-1}{n+1}\right)^2.$$

This determines the value of n as follows:

$$n = \frac{1+\sqrt{F_0}}{1-\sqrt{F_0}}.$$

The determined value n is then used in the *Fresnel* equation again to obtain F with respect to an arbitrary angle δ.

Cook and *Torrance* observed that a colour of the light reflected specularly approaches a colour of a light source according as δ approaches $\frac{\pi}{2}$, since F approaches one. That is, when $F_{\frac{\pi}{2}} = 1$, the colour is not influenced by the object's material. This is an evidence that their model is based on the light property.

On the other hand, a wavelength dependence of F can be obtained by applying a similar way, according to different wavelength [CT82, NIK91].

1.2.2.2 G: Geometrical attenuation factor The geometrical attenuation factor accounts for how much incident light is specularly reflected by microfacets in what directions. If a surface is illuminated and viewed in the normal direction $\vec{N}$ $(\vec{L} = \vec{V} = \vec{N})$, all facets are fully illuminated and visible. For a large incidence angle, however some facets are shadowed and masked by adjacent facets as shown in Figure 2.4.

Masking is the effect where some of reflected light is intercepted by an adjacent facet of V-groove, while shadowing is inversely the effect where a reflecting facet is only partially illuminated. *Blinn* [Bli77] derived geometrical descriptions

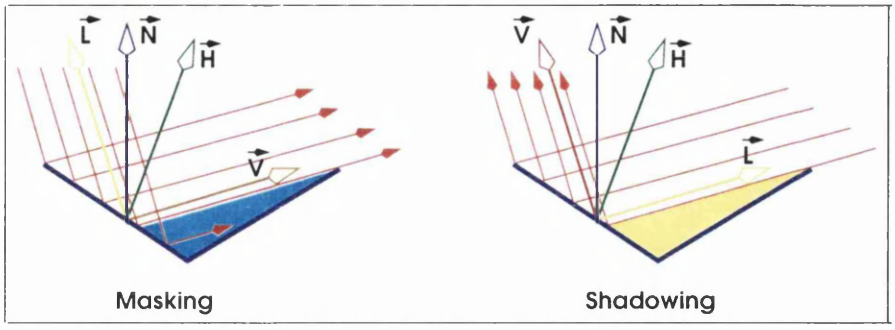


Figure 2.4: Masking and Shadowing

Masking is the effect that some of reflected light is intercepted by the adjacent facet of the V-groove, while shadowing is reversely the effect that the reflecting facet is only partially illuminated.

of the masking G_m and shadowing G_s separately as follows:

$$G_m = \frac{2(\vec{N} \bullet \vec{H})(\vec{N} \bullet \vec{V})}{\vec{V} \bullet \vec{H}}$$

$$G_s = \frac{2(\vec{N} \bullet \vec{H})(\vec{N} \bullet \vec{L})}{\vec{V} \bullet \vec{H}}$$

If incident light is totally reflected to a viewing direction $\vec{V}$, there is no masking and shadowing effect on the facet. So the geometrical attenuation factor $G = 1$. Otherwise, G is dependent on either masking G_m or shadowing G_s. *Cook* and *Torrance* used Blinn's descriptions to obtain the geometrical attenuation factor G, which is the minimum of three values as follows [Bli77]:

$$G = min(1, G_m, G_s).$$

1.2.2.3 D: slope Distribution function The slope distribution function D is a statistical measure for describing what fraction of an incident light is reflected from well-oriented microfacets whose $\vec{n_f}$s are in the direction of the halfway vector $\vec{H}$. Since the microfacets reflect perfectly, the specular reflectance is only dependant on the fraction D of the microfacets having that orientation. There are a number of slope distribution functions in the literature [Lew94].

One of simple functions is the *Gaussian* distribution function [TS67, HB88] as follows:

$$D = ce^{-(\frac{\alpha}{m})^2},$$

where c is an arbitrary constant, $\alpha = \cos^{-1}(\vec{N} \bullet \vec{H})$, and m indicates the surface roughness as a mean square slope of microfacets.

Cook and *Torrance* [CT82] introduced the *Beckmann* distribution function [BS63] for rough surfaces as follows:

$$D = \frac{1}{4m^2 \cos^4 \alpha} e^{-(\frac{\tan \alpha}{m})^2},$$

where $\alpha = \cos^{-1}(\vec{N} \bullet \vec{H})$ and m is a mean square slope of the microfacets. It does not include any arbitrary constants but requires more computation.

Schlick proposed a simpler and less expensive computational function, which has not the exponent term, as follows [Sch94]:

$$D = \frac{\cos^2 \alpha}{((m - r)\cos^4 \alpha + r)^2}, \tag{2.3}$$

where $r = \frac{1}{2m}$, again $\alpha = \cos^{-1}(\vec{N} \bullet \vec{H})$, and m is a mean square slope of the microfacets.

In all of the slope distribution functions, the sharpness of the specular reflectance depends on the surface roughness m. In practice, it varies from 0 (not included) to 0.5 for real surfaces [Sch94]. A small value of m indicates a smooth surface so that the reflection is sharply directional, as shown in Figure 2.5 a), because the microfacet slopes vary only a little from a mean surface normal $\vec{N}$, which can be referred as a specular spike. While large values of m indicate rough surfaces having steep microfacet slopes. The spread of the reflection is wide like a specular lobe, as shown in Figure 2.5 b).

1.3 Discussion

In this section, we have reviewed several shading models. These determine an image intensity value $I_{\vec{V}}$ on a surface or image point, in terms of the three components: diffusely reflected light, specular lobe, and specular spike. They combine functions containing geometric and light properties; for example, a light source vector $\vec{L}$, a surface normal vector $\vec{N}$, a halfway vector $\vec{H}$, and so on.

A combination of these components can simulate some surfaces in reality; for example, a plastic and metallic surface. On the contrary, it is difficult to extract the geometric and light property from an image simulated by such a combination mathematically.

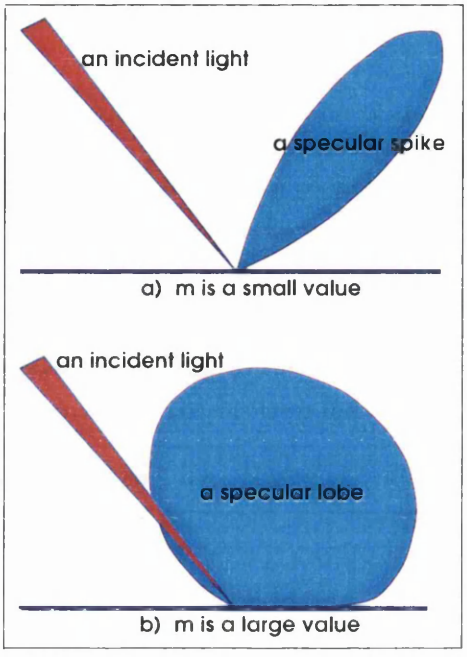


Figure 2.5: An example of a slope distribution function

A small value of m makes the specular reflectance sharply directional, a specular spike, while a large value m produces a specular lobe. These pictures are adopted from [CT82]. When $m = 0.2$, the *Beckmann* distribution function looks like the upper one. In addition, when $m = 0.6$, it looks like the lower one.

For example, we assume that an intensity value $I_{\vec{V}}$ at a particular image point is given by a combination of three components as follows:

$$\begin{aligned} I_{\vec{V}} &= I_d + I_{sl} + I_{ss} \\ &= I_d + I_s \\ &= I_p k_d (\vec{N} \bullet \vec{L}) + \frac{I_p k_s F D G}{(\vec{N} \bullet \vec{L})(\vec{N} \bullet \vec{V})} \\ &= \vec{N} \bullet \vec{L} + \frac{D}{(\vec{N} \bullet \vec{L})(\vec{N} \bullet \vec{V})}, \end{aligned}$$

where the coefficients I_p, k_d, k_s, F, and G are equal to one, D is the slope distribution function represented by the equation (2.3), the vectors are normalised, I_d is a diffuse component (eq. (2.1)), and I_s is also the specular one (eq. (2.2)) of the *Cook-Torrance* model. It is intractable to compute a unique solution of $\vec{N}$, even if we employ a sophisticated *SFS* method and $\vec{L}$ and $\vec{V}$ are given additionally, because we have only one equation with three unknown in $\vec{N}(N_x, N_y, N_z)$. *Bruss* [Bru82] proved that no *SFS* method can yield a unique solution without additional constraints.

This is the reason why most *SFS* methods employ the simplest shading model, referred to as the *Lambertian* reflectance model. It is for this reason that we also will reconstruct faces using single *Lambertian* images.

2 Rendering Polygonal Meshes

Rendering is the overall act that processes from a 3D surface representation of an object to a 2D image projection on a view plane. A rendered image can be obtained by applying a shading model to either every visible surface point, or every interpolated surface point.

The former is usually used by ray tracing or radiosity method, while the latter is used by interpolation methods. In this section, we are interested in rendering methods for the interpolated surface points, since we adopt a mesh representation for human faces in this thesis.

There are three classical ways to render a polygonal mesh in order to simulate smooth surfaces: constant intensity, intensity interpolation, and normal interpolation rendering methods. To understand them in brief, we are going to keep the discussion in 2D.

Figure 2.6 shows both intensity profiles (left) and simulated surfaces (right), which can be obtained by applying each rendering method to a polygonal approximation of an original surface (top).

The original surface is a cylindrical shape, which is approximated by four polygons. Each polygon is also represented by a straight line between two vertices. Arrow marks on the polygons represent either a surface or a vertex normal vector, while the others are normal vectors used by each rendering method.

A point light source is assumed to be illuminating vertically in the plane from top to bottom. In addition, we assume that an image intensity value at an interpolated surface point is obtained by applying the *Lambertian* shading model represented by the equation (2.1). We also assume that the intensity of the light source I_p and the reflectance coefficient k_d are equal to one, because we are most interested in how the image intensity value is obtained by $\vec{N}$. This vector is differently given according to rendering methods.

2.1 Constant rendering

The constant intensity rendering method applies a single intensity to a polygon [Bou70] so that all points over the polyogn have the same intensity value. Therefore, the intensity profile looks like steps as shown in Figure 2.6.

The vector $\vec{N}$ used in this method is a surface normal corresponding to a polygon, as shown on the intensity profile. The simulated surface is similar with

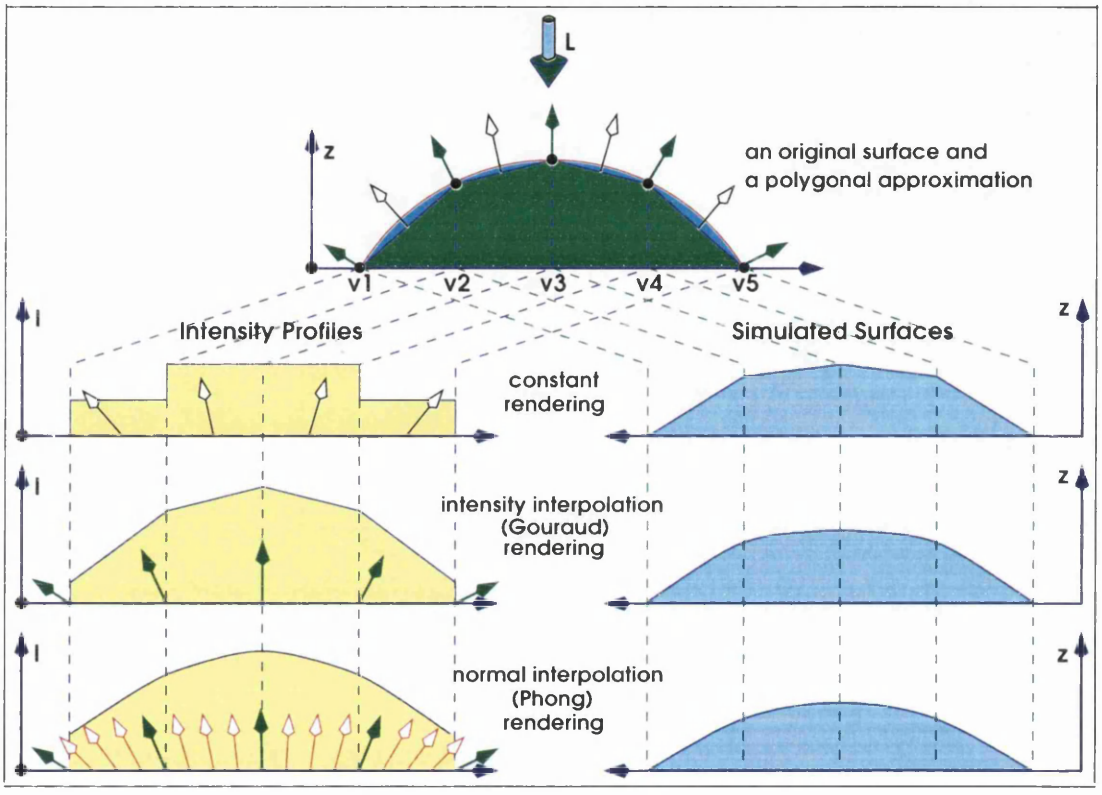


Figure 2.6: An example of rendering a polygonal mesh

Intensity profiles (left) and simulated surfaces (right) can be obtained by applying each rendering way to a polygonal approximation of an original surface (top). The original surface is a cylindrical shape, which is approximated by four polygons. Each polygon is also represented by a straight line between two vertices. Arrows on the polygons represent either a surface or a vertex normal vector, while the others are normal vectors used by each rendering method. A point light source is assumed to be illuminating the surface vertically in the direction from top to bottom.

the polygonal approximation, because each polygon is represented by only a surface normal.

It means that this rendering method does not simulate an original smooth surface accurately but simulates its appearance. Consequently, the simulated surface retains the polygonal structure.

2.2 Intensity interpolation rendering

The intensity interpolation rendering simulates a smooth surface by linearly interpolating the intensity values of two adjacent vertices. This is also known as *Gouraud* rendering [Gou71].

An intensity value of a vertex is obtained by a vertex normal vector. as shown on the intensity profile. It is an average of vectors which are surface normals of polygons sharing the vertex. Now the intensity profile looks linearly smooth, which eliminates the intensity discontinuities of the constant rendering method. However there still exists the intensity disparity between adjacent polygons around shared vertices v_2, v_3, and v_3 on the intensity profile. Generally speaking, this disparity appears as brighter or darker intensity streaks along shared edges than their surroundings, whenever the slope of intensity curve changes. These streaks are called *Mach* bands [Rat72, Pho75].

On the other hand, the intensity interpolation implies the change of surface normals between two vertices. So the simulated surface approximates to the original, as shown in Figure 2.6.

2.3 Normal interpolation rendering

The normal interpolation rendering simulates a smooth surface by linearly interpolating the normal vectors instead of the intensity values of two adjacent vertices. This is called *Phong* rendering [Pho75].

The intensity profile shows an improvement on the intensity discontinuities around the vertices, by means of applying the interpolated normal vectors. They are represented by the arrow marks on the intensity profile.

The surface simulated by this method looks like the original surface, as shown in Figure 2.6. It is more accurate than that of the intensity interpolation rendering. Again, it greatly reduces the *Mach* band effect.

2.4 Discussion

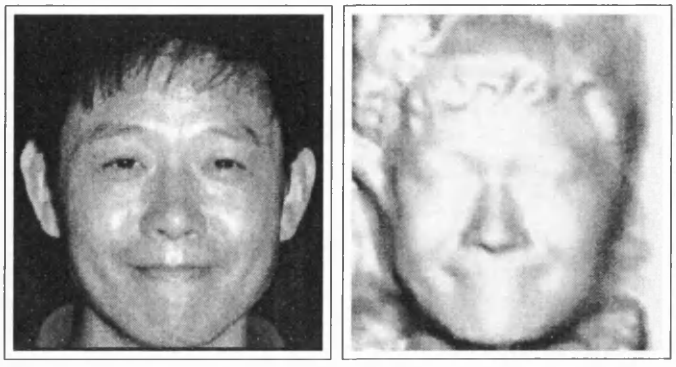


Figure 2.7: A real face photograph and Lambertian image

The left-hand side is a real photograph containing specular components, for example, which are observed on the middle of the forehead and on the tip of the nose. They are intensity discontinuities. On the other hand, the right is a *Lambertian* image rendered by applying the normal interpolation method to a polygonal mesh. The photograph and mesh are adapted from *C3D* of *The Turing Institute*.

The goal of this thesis, as mentioned in *Chapter 1*, is to reconstruct an original face from its single *Lambertian* image using an impoverished face as a model. So we must provide facial images and models for the reconstruction first of all. We would like to utilise real photographs for our purpose, however they usually contain the specular reflectance components as shown on the left-hand side in Figure 2.7.

To be well set up with images and models at the same time, we will use polygonal meshes of original faces, which were adapted from *C3D* of *The Turing Institute*[4]. Now the *Lambertian* face images can be prepared by the normal interpolation method as shown on the right-hand side in Figure 2.7. This *Lambertian* image represents the original face as well as the photograph. Generally, a *Lambertian* image is a fairly realistic approximation for human faces. *Ikeuchi* and *Sato* [IS91] have quantitatively confirmed that human faces are dominated by *Lambertian* reflectance. This suggests that a *Lambertian* image of a face is a satisfactory substitute for its photograph. From this sense, we will use *Lambertian* images as inputs for the face reconstruction in this thesis. On the other hand, the issues about models will be discussed in the next chapter.

[4] *C3D* and *The Turing Institute* are trademarks of the Turing Institute.

3 Shape from Shading

So far in this chapter, we have reviewed the shading models in terms of three reflectance components for a surface point, and then how well the polygonal renderers simulate an original surface using the *Lambertian* shading model. We have discussed how to decide an intensity value on a particular point, assuming that surface and shading properties are given.

In this section, we will explore inverse shading. Assuming that an intensity value and shading property are given, how can we recover the surface property on a particular point?

3.1 A simple example of a surface recovery

The *SFS* method provides one solution to the previous question. In order to explain this, we start with an ideal example in 2D, as shown in Figure 2.8. Given an intensity profile (shading), find the original surface (mountain shape) satisfying a shading model.

We assume that there is a distant light source $\vec{L}$ illuminating from top to bottom and that the intensity profile is a set of intensity values for each pixel. Most *SFS* methods extract surface orientations corresponding to all individual pixels and then recover the original surface using integration.

For example, the arrowmarks in Figure 2.8,which is called a needle diagram [Hor86], can be a set of solutions of surface normals, since they give the same intensity profile as the input. This allows us to get a recovered image illuminated by a different light source. However, the image cannot be viewed in a different direction, because such needle diagrams only provide a tiny subset of discontinuous surfaces, as shown on the left-hand side in Figure 2.8.

Usually the original surface is recovered by applying the integrability of a smooth surface, as shown on the right-hand side in Figure 2.8. Unfortunately, the recovered surface is now too flat to be seen in a differenent view as well, because it uses an over smoothness constraint. This means that *SFS* methods must adopt additional constraints in order to produce a different view. We must, therefore, exploit an alternative method in this thesis.

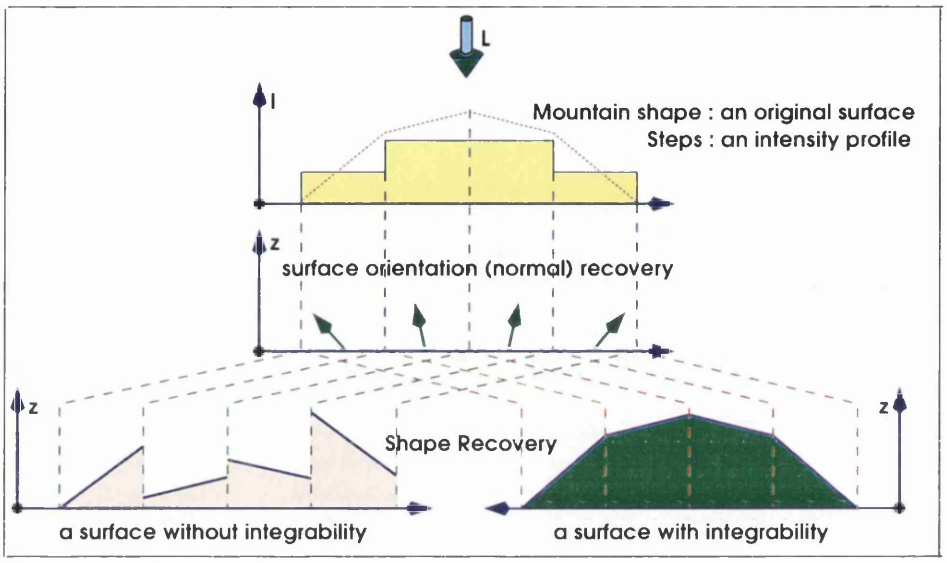


Figure 2.8: A simple example of shape from shading

A *SFS* method is an inverse shading process: given an intensity profile (shading) I, find the original surface (mountain shape) satisfying the image irradiance equation, $I = R(p, q)$ (shading model). Most *SFS* methods extract surface orientations (a needle diagram), middle, for each pixel and then recover the original surface by integrability.

3.2 The constraints often appearing in the SFS

In a computational sense, the common characteristics of *SFS* problems can be formalised as ill-posed problems [PTK85]. It has, however, been proven that they are well-posed in special cases such as at a singular point [Bru82, DS81]. However for a typical *Lambertian* image, a unique surface does not always exist. There may exist an infinite number of possible corresponding surfaces. One means of solving this ill-posed problem is to reformulate it in terms of a well-posed one by employing additional constraints.

Many researchers have investigated these additional constraints. Due to their efforts, a large number of significant improvements in the *SFS* problem have been achieved [Hor75, IH81, HB86, Pen86, FC88, ZC91, BP92b, LK93, OD93].

The constraints employed by most *SFS* methods are

- a *brightness* constraint, which enforces an reconstructed image to be close to the image under analysis [HB86],

- a *smoothness* constraint, which enforces a reconstructed surface to be smooth to ensure the stabilisation and convergence of the recovery process [IH81],
- an *integrability* constraint, which enforces a reconstructed surface to have partial derivatives [FC88], and
- an *intensity gradient* constraint, which enforces the intensity gradients of a reconstructed image to be close to those of the input image [ZC91].

3.3 Shape from shading tree

Although there are a number of conventional *SFS* techniquies proposed so far in the literature, they can be demarcated into several groups as shown in Figure 2.9 [ZTCS94, Pen90].

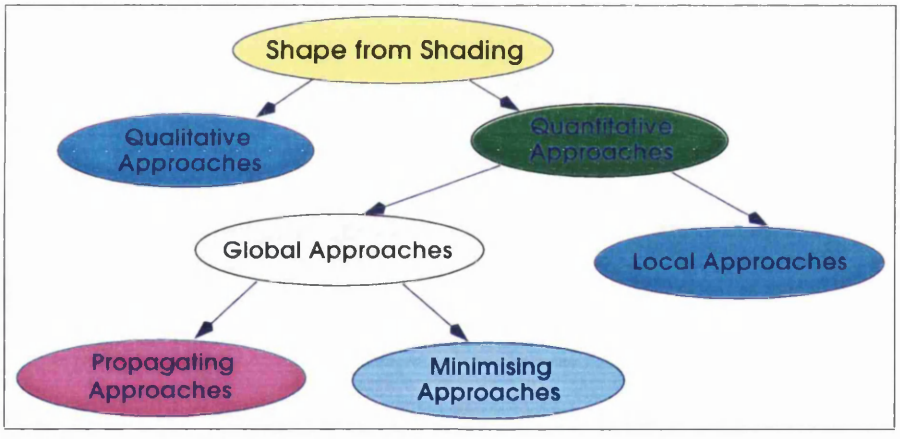


Figure 2.9: Shape-from-shading tree

> Qualitative approaches qualitatively extract 3D shape properties from photometric invariants in an image, such as curvature, convex, and concave. Quantitative approaches determine the quantitative properties of an original surface, such as depth, normal, and slant/tilt. Local approaches recover shape from shading in the vicinity of a surface point. Propagating approaches recover shape by propagating the initial shape information to the whole image. Minimising approaches find acceptable solutions by minimising a cost function.

Most approaches have focused on quantitative methods, but some of them require more qualitative approaches. That is, they determine the qualitative

properties of an original surface such as a convex, concave, saddle, and hyperbolic point, and so on.

Koenderink [KvD80, KvD82, KvD84] proposed that the 3D shape of an iso-brightness curve in an image is determined by parabolic points on the surface, and that an intensity on the parabolic curve is a local minimum or maximum.

Pentland [Pen86] accounted for the qualitative surface type with spherical points such as convex, concave, saddle, planar, cylinderical points. In addition, Dupuis [DO92] used the fact that qualitative curvature of a surface is given at points of brightest image intensity.

On the other hand, quantitative approaches determine the quantitative properties of an original surface. Most of the recovered properties can be represented by one of the following [Hor90, Bru88]:

- a contour map,
- a depth map $z(x, y)$,
- a needle diagram $\vec{N}(N_x(x, y), N_y(x, y), N_z(x, y))$,
- a gradient space $(p(x, y), q(x, y))$, or
- a slant and tilt space $(\rho(x, y), \tau(x, y))$,

where (x, y) means a point (x, y) in an image.

The contour map is a set of equal height curves of a reconstructed surface. The depth map is a collection of relative heights. The needle diagram is a set of unit normal vectors. The gradient space is a collection of the rate of change of heights. The slant and tilt space is a set of the angular version of the gradient space.

Furthermore, the quantitative approaches can be divided into global approaches and local approaches. Global approaches determine the original surface by either propagating known shape information at a singular point to the whole image [Hor75, OD93, BP92b] or minimising a cost function related to brightness errors according to the whole image [IH81, HB86, Hor90, ZC91]. Local approaches determine the original surface in a small neighbourhood of an image point [Pen86]. The next subsections review several of these quantitative approaches starting with the propagating approaches. It is important to understand the characteristic strip method because it was the first to address the *SFS* problem in image analysis [Hor86].

3.3.1 Propagating approach

3.3.1.1 Characteristic strip method Assuming that $z(x, y)$ is a smooth surface so that the first and second partial derivatives exist everywhere, then a depth map $z(x, y)$ can be computed by solving the image irradiance equation, $I = R(p, q)$, where I is an intensity value, $R(p, q)$ is a shading model.

Horn [Hor75] proposed a set of five ordinary differential equations using the image irradiance equation as follows:

$$\begin{aligned} dx &= R_p ds, & dy &= R_q ds, \\ dz &= p dx + q dy, & & \\ dp &= I_x ds, & dq &= I_y ds, \end{aligned}$$

where ds is a small step along a solution curve. Starting from a singular point, the reconstruction process now turns into solving these equations by propagation along a characteristic curve. This produces a series of the small steps ds's. Let us verify this process with a singular point in an image, assuming that it has an initial surface information $(x_0, y_0, z_0, p_0, q_0)$. For a small step $(dx, dy) = (R_p ds, R_q ds)$ from the singular point (x_0, y_0) to a direction of ds, we want to obtain a solution $(x_1, y_1, z_1, p_1, q_1)$. It is clear that $x_1 = x_0 + dx$, $y_1 = y_0 + dy$, and $z_1 = z_0 + dz$, where dx and dy are already given. The change of height dz can be obtained from the third equation, so $dz = p_0 dx + q_0 dy$.

For the next small step, p_1 and q_1 are prepared by $p_1 = p_0 + dp$ and $q_1 = q_0 + dq$ respectively. From the last two equations, $dp = I_x ds$ and $dq = I_y ds$. Therefore $p_1 = p_0 + I_x ds$ and $q_1 = q_0 + I_y ds$.

This process propagates depth and surface normal information outwards, starting from an initial point in an image[5]. The direction of the propagation is decided by $(R_p ds, R_q ds)$ and $(I_x ds, I_y ds)$. In other words, a step (dx, dy) taken in an image is parallel to (R_p, R_q) of the gradient space, while a step (dp, dq) taken in the gradient space is parallel to (I_x, I_y) of the intensity gradient[6].

A particular set of solutions consecutively determined by this process forms a curve in the image space, which is called a characteristic strip. It can determine a space curve as well, because it has surface normals at all points on it.

One drawback of this approach is that the propagation direction is distorted, if the image is noisy. Moreover, the strips cumulatively deviate from their ideal paths as the computation progresses. In bad cases, the adjacent strips may cross and spread too far apart. However it is possible to avoid crossing and spreading

[5]The initial point can be provided by singular points or occluding boundaries [Hor86].

[6]In practice, the step is usually taken in the direction of the steepest descent path.

problems using certain neighbouring rules [Hor75, Hor90]. This modified method interpolates new strips when the existing ones separate too far, and delete old ones when they are too close to each other.

3.3.1.2 Optimal control method Dupuis and Oliensis reformulated the *SFS* problem as an optimal control problem [Oli91b, DO92, OD93]. By assuming twice differentiable height, isolated singular points, and nonzero curvature at singular points, they suggested that the optimal movement control principle could lead to a solution without the additional constraint used in minimising approaches.

The solution surface is reconstructed from the singular points to the boundaries in an outward propagating manner. In practice, however, the singular points lead to a triple local ambiguity; that is, is the shaped point convex, concave, or saddle-shaped?

Bichsel and Pentland [BP92b] simplified their approach by employing the minimum downhill principle, which removes the ambiguity of singular points. The propagation of depth information is as follows:

- The depth information is only passed to paths that are farther away from the light source.
- Among all the possible paths, choose the steepest descent path that is closest to the light source.

Assuming that the surface is continuous in the direction ϕ and that the depth information is propagated from a point $(x, y, 0)$ toward (x, y, ϕ), the depth z at a possible path ϕ is described by:

$$z^{t+1}(x,\ y,\ \phi) = z^t(x,\ y,\ 0) - dz(\phi)$$

where t denotes the time step, dz is the change of depth in the direction ϕ. Among the possible paths and the previous path, the depth z closest to the light source is selected by:

$$z^{t+1}(x, y, 0) = max(z^{t+1}(x, y, \phi), z^t(x, y)).$$

This approach has the drawback that singular points must be given and the surface should be continuous in the downward direction ϕ. Furthermore it has difficulty with multiple singular points. If they do not have the same depths, this approach will have trouble initialising their depths.

We have implemented the method proposed by Bichsel and Pentland [BP92b] to give an example of the propagating approaches as shown in Figure 2.10. The left-hand side is a *Lambertian* image produced by applying the normal

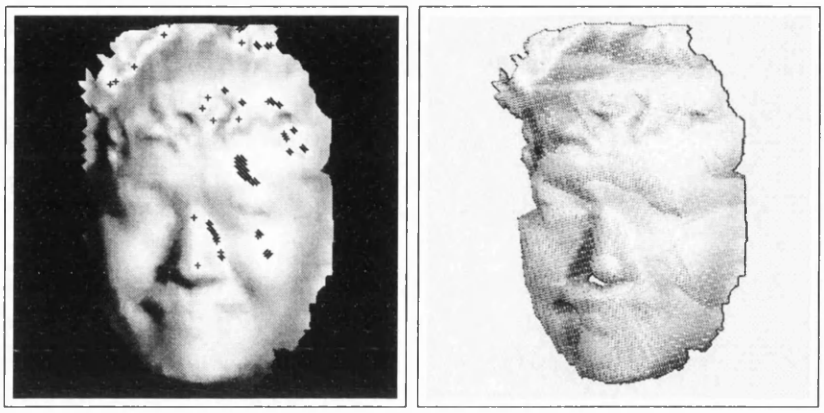


Figure 2.10: An example of propagating approach

The left-hand side is a *Lambertian* image synthesized by applying the normal interpolation renderer to a polygonal mesh with $\vec{L}(0.40, 0.21, 0.89)$, $I_p = 255.0$, and $k_d = 1.0$. The right-hand side image shows the *Lambertian* image of a reconstructed face at the 11^{th} iteration, which was reconstructed by Bichsel and Pentland's method based on a minimum downhill principle.

interpolation renderer to a polygonal mesh with a light source $\vec{L}(0.40, 0.21, 0.89)$. The black cross marks in the input image are singular points, which assumed the same heights. The right-hand side image shows the *Lambertian* image of a reconstructed face at the 11^{th} iteration. As can be seen, it is noticable that the shadow part and background of the input image were not recovered.

3.3.2 Minimising approach

Propagating approaches compute an exact depth solution $z(x, y)$, provided that the image irradiance equation $I(x, y) = R(p, q)$ is correct. On the contrary, however, minimising approaches assume that the image irradiance equation is not correct. That is to say, it has an error term, which is due to shading modelling errors such as incorrect estimates of light source direction, surface reflectance, camera calibration, and so on.

This approach computes an approximate solution whose rendered image is close to the input image. This is based on minimising a cost function[7], which is a combination of the constraints, including one for brightness $||I(x,y) - R(p,q)||^2$, which make computation stable and convergent.

Ikeuchi and Horn [IH81, HB86] employed a surface smoothness constraint $p_x^2 + p_y^2 + q_x^2 + q_y^2$, which enforces a solution surface to be smooth. Horn pointed out one weakness with this. The solution walks away from $I(x,y) = R(p,q)$ when the correct solution is used as an initial condition; for example, when the input image contains singular points [Hor90, Oli91a]. This suggests that the initial condition must be selected carefully for the minimising approach.

Since shading is related to the change of surface orientations, in areas where image intensity changes rapidly, the surface is not smooth. Zheng and Chellappa [ZC91] employed an intensity smoothness constraint $(R_x - I_x)^2 + (R_y - I_y)^2$ instead of using the surface smoothness one. They ensure that the intensity gradient of a solution image should be close to that of the input image. However, these constraints result in a solution image, which looks too flat.

It is noticable that the constraints discussed so far are focused on minimising the intensity error between the input image and a recovered one. Moreover, a reconstructed surface obtained by applying only those constraints may give unacceptable spatial trajectories as shown on the left-hand side in Figure 2.8. These trajectories take the solution away from the original surface.

To minimise the surface trajectories, Horn [Hor90] employed an integrability constraint $(z_x - p)^2 + (z_y - q)^2$, which enforces a reconstructed surface $z(x,y)$ to have partial derivatives z_x and z_y. They should be close to the computed $p(x,y)$ and $q(x,y)$. Alternative integrability constraints are described in [HB86, FC88]. The variation in height z is determined by minimising a cost function in an iterative manner.

Another drawback of the minimising approach is that it requires a huge number of iterations in order to obtain an acceptable result. Szeliski [Sze91]

[7] In standard regularisation theory, the ill-posed problem of finding a solution z from data y,

$$Az = y$$

requires minimising a cost function,

$$||Az - y||^2 + \lambda ||Pz||^2,$$

where A is a linear operator, $||Pz||$ is a stabilising function, and λ is a weighting parameter that controls the compromise between the level of regularisation of the solution and its closeness to the data y. The cost function has to be both close to the data and regular by making the constrained term $||Pz||^2$ small.

provided a faster solution using conjugate gradient descent.

Amongst a variety of the minimising approaches, we have implemented a method proposed by Zheng and Chellappa [ZC91]. Their cost function is

$$\int\int ((I(x,y) - R(p,q))^2 + (R_p p_x + R_q q_x - I_x)^2 + (R_p p_y + R_q q_y - I_y)^2 + \mu((p - z_x)^2 + (z_y - q)^2))dxdy,$$

where μ is a weighting parameter.

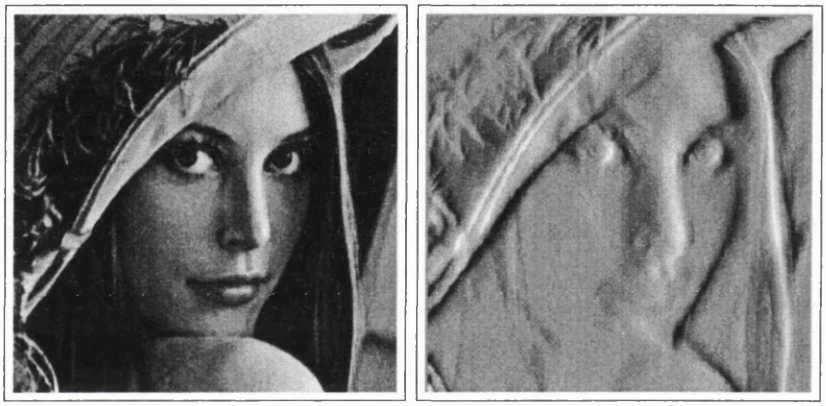


Figure 2.11: An example of minimising approach

The left picture is a standard image regarded as a *Lambertian* image with a light source $\vec{L}(0.78, 0.16, 0.61)$, $I_p = 255$, and $k_d = 1.0$. The right one is a reconstructed image obtained by applying Zheng and Chellappa's method with 2000 iterations and $\mu = 1$, which was synthesized by the same shading model as the input. This looks too flat like a face in a coin.

Figure 2.11 shows a result obtained by applying their method through 2000 iterations. The left picture[8] is a standard image regarded as an approximation to a *Lambertian* image with a light source $\vec{L}(0.78, 0.16, 0.61)$, $I_p = 255$, and $k_d = 1.0$. The right one is a reconstructed image obtained by 2000 iterations with $\mu = 1$, which was synthesized by the same shading model as the input.

At this point, it should be said that the reconstructed image looks too flat

[8]This image is adapted from the *Computer Science Department*, University of Central Florida, Orlando, USA.

due to the over intensity smoothness constraint. In general, any method using the smoothness constraint produces a flat solution similar to their result.

3.3.3 Local approach

While the global approaches recover the shape information over a whole image either in a propagating or minimising manner, the local approaches attempt to recover a surface using assumptions about surface shape in a small neighbourhood of a point.

Under the assumption of *Lambertian* shading, Pentland [Pen86] firstly proposed a solution for local surface types qualitatively and orientations quantitatively. He suggested five surface types corresponding to their curvatures: a planar, cylinderical, convex, concave, and saddle surface. For an image point, these types can be locally determined by its second derivatives d^2I as follows:

- a plane satisfies that $d^2I = 0$ in all directions,
- a cylinder satisfies that $d^2I = 0$ along one directions, and
- if $d^2I \neq 0$ in all directions, then the image point is a convex, concave, or saddle surface.

In addition, a surface orientation is recovered by matching the spherical surface derivatives with the brightness derivatives in an image. This is described by the slant and tilt representation, where `tilt` is defined as the angle between the projection of the surface normal on the image plane and the x-axis, and `slant` is the angle between the surface normal and the direction toward the viewer.

For a surface normal $N(X_n, Y_n, Z_n)$, the slant σ is the depth component of it and is equal to $\cos^{-1} Z_n$. Furthermore, the z-component Z_n is approximately estimated by

$$\begin{aligned} \cos\sigma &= Z_n \\ &= \frac{\frac{1}{R}}{\sqrt{|\frac{\nabla^2 I(x,y)}{I(x,y)}| - \frac{1}{R^2}}}, \end{aligned}$$

where $\nabla^2 I(x,y) = I_{xx}(x,y) + I_{yy}(x,y)$ is the *Laplacian* operator of image intensity and R is the radius of spherical patches.

The tilt τ, which is the image plane component of the surface normal, is as follows:

$$\tan\tau = \frac{Y_n}{X_n},$$

where the tilt direction $\frac{Y_n}{X_n}$ is estimated as the maximum response of the gradient direction of the slant field.

His approach is useful to recover the shape in the absence of the initial conditions such as singular points and occluding contours. With an assumption of equal-curvatures, however, it is not possible to determine the surface shapes uniquely. With the second derivative of image intensity, it may give ambiguous solutions; such as a convex, concave, and saddle surface. Another drawback is that he used the second derivative of image intensity to determine the surface types and orientations. The use of the second derivatives makes this approach very sensitive to noise.

On the other hand, under the same spherical assumption, Lee and Rosenfeld [LR85] recovered a solution surface in terms of slant and tilt in the light source coordinate system. They used only the first derivatives I_x and I_y, which is different from the second derivatives d^2I used by Pentland. Under the *Lambertian* surface, they proposed that the slant σ and tilt τ of a surface normal at an image point be obtained as follows:

$$\tan\tau = \frac{I_y \cos\tau_l - I_x \sin\tau_l}{I_x \cos\tau_l \cos\sigma_l + I_y \sin\tau_l \cos\sigma_l},$$

$$\cos\sigma = I,$$

where I is the normalised intensity, and σ_l and τ_l are the slant and tilt of a light source respectively. Empirically, we found that their approach strongly depends on a light source. In particular it only works properly for the light source $\vec{L}(0,0,1)$.

We have also implemented a method proposed by Lee and Rosenfeld [LR85] to further investigate the strengths and weaknesses of the local approaches.

Figure 2.12 shows a result obtained by applying their method. The left-hand side is a *Lambertian* image generated by applying the normal interpolation renderer to a polygonal mesh, together with a light source $\vec{L}(0,0,1)$, $I_p = 255$, and $k_d = 1.0$. The right-hand side shows a reconstructed image, which was synthesized by the *Lambertian* shading model with $\vec{L}(0.55, 0.18, 0.82)$. The reconstructed image looks right.

However it should be said that the height information is not correct, because it was obtained by the formula, $\cos\sigma = N_z = I$, which means that a height component is equal to an intensity value. For example, if we assume that the tip of the nose has the same intensity value as a point on the mouth, then they will have the same height.

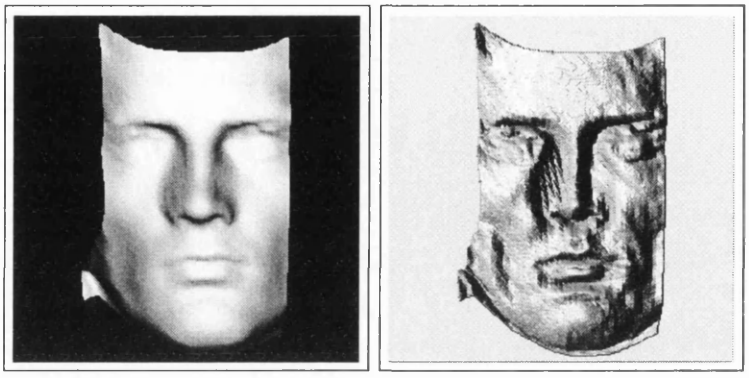


Figure 2.12: An example of local approach

The left-hand side is a *Lambertian* image generated by applying the normal interpolation renderer to a polygonal mesh, together with a light source $\vec{L}(0,0,1)$, $I_p = 255$, and $k_d = 1.0$. The right-hand side shows a reconstructed image obtained by applying Lee and Rosenfeld's method, which was synthesized by the *Lambertian* shading model with $\vec{L}(0.55, 0.18, 0.82)$.

3.4 Discussion

We have discussed many approaches to the *SFS* problem. We demonstrated three quantitative methods. Table 2.1 compares these approaches, where time units are relative values[9] and the input image and the result are shown in Figure 2.10, 2.11, and 2.12 respectively.

Author	Approach	Figure	Time (sec)	Iterations	Problems
Bichsel & Pentland	Propagating	2.10	30	11	singular pts
Zheng & Chellappa	Minimising	2.11	240	2000	flat/time
Lee & Rosenfeld	Local	2.12	5	1	wrong depth

Table 2.1: Comparing the three *SFS* approaches

The propagating approaches require the initial shape information, such as a singular point or occluding boundary, in order to start finding a solution.

[9]The machine used is SPARCstation 20 with two 75Mhz SuperSparc processors and 256 Mbytes of RAM

Bichsel and Pentland proposed an optimal control method, which required at least a singular point. We have applied their method to another input image in Figure 2.11, which did not have singular points. It failed.

The minimising approaches, as we mentioned before, find an acceptable surface by means of minimising a cost function, which includes additional constraints. For example, brightness, smoothness, and integrability constraints were all we used. These approaches require a number of iteration to obtain an acceptable result, as shown in Table 2.1. In addition, the smoothness constraint makes the result too flat.

The local approach is to find locally the original shape of an input image, with a local shape assumption. Lee and Rosenfeld employed a spherical assumption. Their method is faster than the others, but it may generate wrong local depth information.

4 Face Reconstruction: A Survey

We have now reviewed three main issues; shading models, rendering models, and *SFS* approaches. In this section, we survey methods for reconstructing 3D human faces.

Many approaches have been proposed so far. The face model is usually reconstructed from facial images or range data that are obtained using special hardware equipment. These techniques are exploited in various application fields including model-based coding [Cla95], facial animation [PW96], facial surgery simulation [KGC96], and forensic identification [MC96].

Reconstruction methods can be divided into four groups according to the data they used. The first group uses stereo images from a pair of ordinary cameras [SU94]. The second group is statistically based on a huge set of facial shape components [AGR95, AGR96]. The third group uses a pair of orthogonal images, which are a front and side facial image, and a prototype face model [AC91, ASW93, HY96, Vet96]. Finally, the fourth group uses a set of depths, so called range data, captured by a laser scanner [DY88, NHRD90, WT91, LTW93].

All of these methods have a main drawback; that is, a physical subject is necessary to obtain a pair of images or range data from an individual face.

4.1 Stereo images

C3D is a novel system for reconstructing a 3D individual face from stereo image pairs, developed by the *Turing Institute*[10] [SU94]. It consists of five modules: image acquisition, stereo matching, photogrammetric and calibration, 3D reconstruction, and visualisation.

In the image acquisition process, the stereo image pairs are captured under textured light to ensure a uniform spatial distribution of high frequency texture and under normal light to render a photo-realistic face. The left-hand side in Figure 2.13, which was adapted from [SU94], shows a pair of stereo images captured under textured illumination.

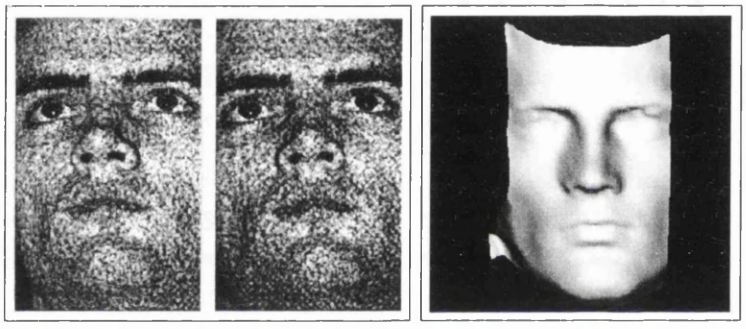


Figure 2.13: An example of reconstructing from stereo image pairs [SU94]

It employs a stereo matching algorithm to determine the displacements between the stereo image pairs. The stereo image matching is achieved by the pyramid of eight image resolutions, starting from the coarsest resolution to the finest resolution.

Photogrammetric and calibration analysis is applied to the displacements in order to obtain a depth map of the original face. The depth map represents height values at every pixel.

The 3D reconstruction converts the depth map into a polygonal mesh through range image operations; *Delaunay* triangulation [FP93, Kum96] and mesh simplification [Eri96]. Finally the visualisation is the process of displaying the reconstructed mesh on the screen, rendered by a graphical method. The right-hand side in Figure 2.13 shows a reconstructed face rendered by the normal interpolation method.

[10] *C3D* and *The Turing Institute* are trademarks of the Turing Institute.

This method relies upon the subject being available to generate the stereo image pairs. If only a single image is available; for example, a photograph of a fire victim's face or a missing child's, it cannot be applied.

4.2 Facial shape components

Atick and others [AGR95, AGR96] have proposed a statistical method for reconstructing a face. This is based on a set of facial shape components. There are some interesting points on their method because it reformulated the minimising approach for *SFS* problem. Here we briefly introduce it.

They suggested that objects can be classified into classes according to their shape. The shape space within each class can be parameterised by principal component analysis [Jol86, SK87, KS90]. Now the *SFS* problem then becomes equivalent to estimating a small number of principal components in an image and an object class.

Atick *et al.* analysed hundreds of 3D scanned male heads as a class and derived a *meanhead* and a set of *eigenheads*. The meanhead is an average head shape from the scanned heads and the eigenheads are a huge set of standardised variations of each head from the meanhead. Figure 2.14 adapted from [AGR95] shows them. The upper left-most corner in the left box represents the meanhead, while the others are a subset of eigenheads. The right box also shows two results; the first columns are *Lambertian* images as input to their method and the second ones show the reconstructed heads, which were obtained by adding a linear combination of the eigenheads to the meanhead, given a specified error boundary.

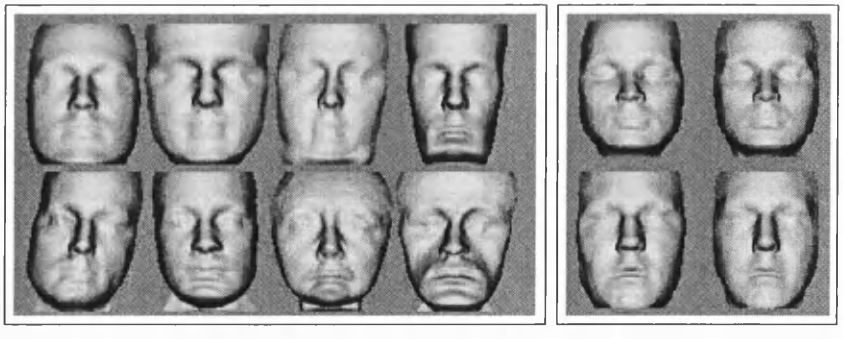


Figure 2.14: An example of reconstructing from facial shape components [AGR95]

This approach uses cylindrical coordinates to represent an out-of-sample head, $r(\theta, h)$, by both the meanhead $r_0(\theta, h)$ and a combination of eigenheads $\phi_i(\theta, h)$:

$$r(\theta, h) = r_0(\theta, h) + \sum_i a_i \phi_i(\theta, h),$$

where θ is angle, h is height, and a_i is a principal component coefficient.

The eigenheads $\phi_i(\theta, h)$ represent an empirically derived set of statistical regularities in the 3D head database. Again, they produce a set of low-dimensional representations. Thus the required number of coefficients depends on how accurately the face shape is reconstructed.

The reconstruction of the head $r(\theta, h)$ from an image now becomes the problem of determining the coefficients a_i's. Atick et al employed a cost function to minimise brightness constraints as follows:

$$\int\int (I(x_0 + r\sin\theta, h) - R(\vec{L}, \mathbf{a}))^2 d\theta dh$$

where $I(...)$ is an estimated intensity in the cylindrical coordinates and $R(...)$ is a reflectance function with respect to a light source $\vec{L}$ and a set of coefficients a_i's.

This approach suffers from several major drawbacks in its use of the eigenheads. It strongly depends on the quality of the eigenhead surfaces as a basis of the database used. If we use poor eigenheads produced by a noisy ensemble of 3D heads, the reconstruction may not be acceptable. The storage requirement for the database is huge if the eigenheads are numerous. Furthermore, if we try to reconstruct a face that does not belong to the distribution of the eigenhead set, the quality of the resultant face may be poor, because it depends upon the meanhead.

4.3 Prototype model modification

A face model corresponding to an individual can be obtained by modifying a prototype model in the form of a polygonal mesh, using a front and a side image of the face [ASW93, HY96]. The left-hand side box in Figure 2.15 shows a modified prototype model imprinted on the orthogonal images, which was adapted from [ASW93].

The modification is achieved by rearranging a set of displacement vectors, which is obtained from the correspondence of facial features between the prototype model and the two orthogonal face images. However, this method requires that feature match between the model and the images should be carefully carried

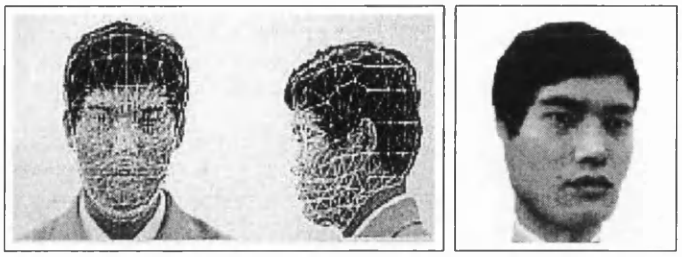


Figure 2.15: An example of reconstructing from a prototype model [ASW93]

out. Another drawback is that, if tilted image pairs are under analysised, this method may not work, because it is endowed with a pair of orthogonal images. In addition, it is assumed that the human face is right-left symmetrical.

Texture mapping is generally applied to enhance the realism of a face. Even when the modified geometry of the face is coarse, a texture mapped face can be seen as a real one. The right-hand side box in Figure 2.15 shows a texture-mapped face image, which was adapted from [ASW93]. On the other hand, to generate more accurate individualised face models, other facial views may be added to provide further information on the other side of the face.

4.4 Range data

Facial dimensions can be obtained by a range finding technique, which scans the face of an individual with a large set of light-lines. This is just a coarse version of *C3D* with textured illumination. It provides both range and texture data simultaneously [TK92, CS94]. The range data is a collection of unorganised and incomplete 3D points from an object surface, and hence it is view-variant; that is, the surface can only be viewed in limited directions. The left-hand side in Figure 2.16, which was adapted from [TK92], shows a shaded example of range data of a face.

The view-variant problem naturally raises the question of how to reconstruct such data [DY88, Mur91, TV91]. Amongst different possible reconstruction schemes, surface triangulation is almost always used, because of its computational efficiency.

A reconstructed surface in the form of a 3D polygonal mesh can be visualised from any view points with constant accuracy. The right-hand side in Figure 2.16 shows a shaded image of a reconstructed face. However, in general, a reconstructed surface from range data consists of a huge number of polygons.

Figure 2.16: An example of reconstructing from range data [TK92]

Such a mesh is expensive to store, transmit, render, and even is awkward to edit. Mesh simplification can provide a simple version to deal with these problem [SZL92, Hop94, EDD+95, KT96]. Polygons in a simplified mesh are formed in such a way that where the surface details are complex, the polygons are small, while large polygons are generated where the surface is featureless.

5 Discussion

In this chapter we have looked at shading models, *SFS* methods, and face reconstruction techniques. The shading models are based on the *SFS* methods. Although a number of methods have been proposed, they suffer from some problems. The reconstructed surfaces are often too flat as shown in Figure 2.11 and often described by wrong depth information as shown in Figure 2.12. This suggests that we need another assumption on surface shape. This thesis proposes an internal model of human faces in the form of polygonal meshes, which are very impoverished.

When only photographs are available it is still possible to reconstruct the 3D appearance of a face if there is also a model which could be referenced. For example, if a single photograph is available, can we reconstruct a fire victim's face using a face reconstuction method introduced in this chapter? The answer is no, because

- the stereo and the modification approach require a pair of images,

- the component approach requires that the preburn face belongs to a set of samples, and
- the range approach requires range data.

However, our method is suitable for reconstructing a burn face.

In the next chapter, we will develop tools to simulate the impoverished faces from the originals. They will be used as the internal models in our reconstruction method.

Chapter 3

A Representation of Prior Knowledge

In the fields of both computer vision and graphics, the geometric modelling or representation of a 3D object is an important problem. There is no one way to model it completely, because it depends on the application of the model [PW96].

In this thesis, we need to model as approximation of the original surfaces of human faces. As a planar approximation, we employ a triangulated mesh which consists of a large number of triangles. How many triangles do we need to represent a face? We have no doubt that the more triangles are used, the more detailed face is available [FvDFH90, BG92, FvDF+94, HB94, Wat94].

However our purpose in this chapter is not to approximate human faces in detail, but to impoverish a face model as much as possible in terms of triangles. This impoverished face[1] will play an important part as `prior knowledge`, when we reconstruct the original face from a single face image in the following two chapters.

In this chapter, we will discuss the issues of impoverishing an original face. It starts with the reason of face impoverishment, the attributes of faces, and then addresses the *Meducer* as a tool for impoverishing faces and the *Meditor* which is a tool for editing face meshes. Figure 3.1 shows an example of impoverished levels of a face created by the *Meducer*.

[1] In this thesis, a *face* is referred as a (polygonalised) *face model* or a *face mesh* without discrimination hereafter.

1 Why Do We Impoverish a Face?

Let us imagine that a sculptor has a cubic-shape stone and a face photograph. First of all, he may cut corners of the stone to obtain an ovoid shape. And he carves the features of the face coarsely. And then he refines them again and again. In doing so, as we argued in *Chapter 1*, he probably uses a geometry-based internal model for the face. We can regard a sculptured degree at a point of time as a representation of his internal model (prior knowledge). Therefore we can say that he reconstructs the original face in a photograph, starting from a very coarsely sculptured face such as a cubic-shaped one. Our face reconstruction method is very similar to his in the sense that we reconstruct the original model of a face photograph using prior knowledge.

In its development, we represent prior knowledge in terms of a sculptured face at a point of time, say, an impoverished face. Therefore the impoverished face must contain the prior knowledge of the original face geometry to some degree. Of course, there are many ways to represent prior knowledge about human faces [Par82, Wat87, LTW93, ASW93, PW96]. For example, the meanhead discussed in *Chapter 2* is a representation of prior knowledge. However, it has too much information, because it can be still recognisable as a person's face. Our purpose is to reconstruct the original face in an image using prior knowledge as poor as possible.

How much information about the original face is sufficient for our method? To answer it, we decided to impoverish the original in order to obtain a coarsely sculptured face. As an extreme case, we believe that it is possible to reconstruct the original face from a single face image, starting from a potato-shaped face such as that shown on the upper middle picture in Figure 3.1. The bottom right picture is a shaded image of an original face and the others show differently impoverished levels of the face. They are 90^{+}%, 80%, 60%, 40%, and 20% impoverished using the *Meducer*. Especially, the top left-hand side is 90% impoverished and then distorted using the *Meditor* to be more impoverished.

When only a single photograph is available, it is possible to reconstruct the original face if there is also an impoverished face. For example, our method is suitable for reconstructing burn faces such as that shown on the top left-hand side in Figure 3.1.

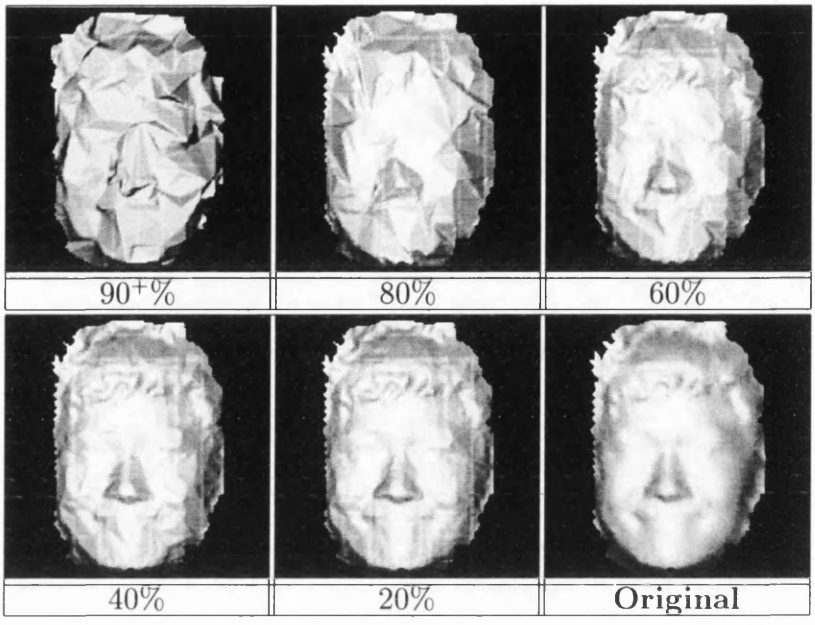


Figure 3.1: An example of impoverished levels of a face

The bottom right picture is a shaded image of an original face and the others show differently impoverished levels of the original face. They are $90^{+}\%$, 80%, 60%, 40%, and 20% impoverished using the *Meducer*. Especially, the top left-hand side is 90% impoverished and then distorted using the *Meditor* to be more impoverished.

2 Attributes of Faces

The *Meducer* and *Meditor* treat face meshes consisting of a large number of triangles. Before we discuss them in detail, the general attributes of face meshes are mentioned in this section.

2.1 Why triangle representation?

Our face models describe the surface rather than the inside of human faces, which is approximated by a polygonal mesh as shown in Figure 3.2. It should be said that we adapted the original faces, used in this thesis, from *C3D* of *The Turing Institute*[2].

[2] *C3D* and *The Turing Institute* are trademarks of the Turing Institute

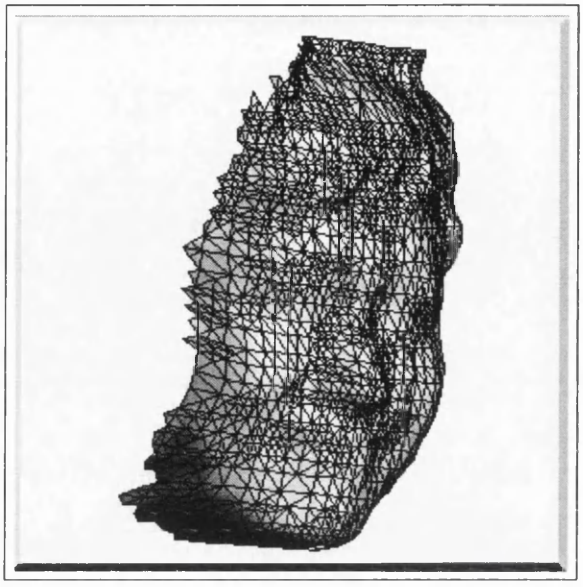


Figure 3.2: A polygonal face model as the original one in Figure 3.1

To build a polygonal approximation, polygons should be tangential to surface patches piecewisely, that is, they should be flat. However there is no way to guarantee that the vertices forming a polygon are coplanar when it is specified with more than three vertices. As shown in Figure 3.3, for example, it is not possible that four non-equal height vertices are coplanar. The vertices V_2 and V_4 make a *valley*, and V_1 and V_3 make a *ridge*.

Therefore we should confirm the vertices to be placed on a plane in order to adapt a polygonal approximation. This confirmation takes a considerable amount of unnecessary time for processing each polygon. Fortunately there is a way to guarantee that the vertices on a polygon are coplanar. This is done by adopting a triangle representation in the form of polygons with three vertices. Accordingly our faces will be approximated by polygonal meshes generated from triangles. Each triangular plane is now piecewisely tangential to the surface patch and the vertices on it are definitely coplanar.

2.2 Data structures

There are four main data structures needed for manipulating the geometric information of face models as shown in Figure 3.4. These data structures have their own attributes.

A face model consists of three pointers `vhead`, `phead`, and `ehead`, which point to a linked list of vertices, polygons, and edges respectively.

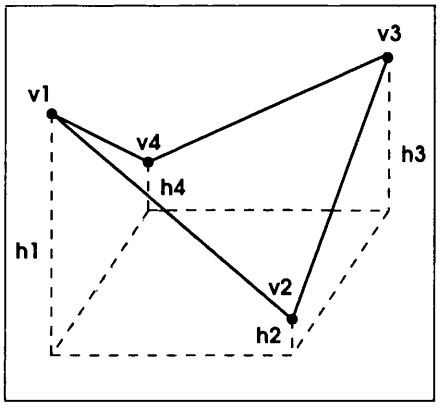


Figure 3.3: Non planar vertices

It is not possible to draw a planar quadrilateral having four non-equal height vertices. Provided that vertices V_1, V_2, V_3, and V_4 have different heights one another, they may contain a valley, V_2V_4 or a ridge, V_1V_3.

2.2.1 Vertex

A vertex is a structure consisting of seven fields as follows;

- The `vtype` indicates whether a vertex referred to is a *boundary* or *surrounded* vertex. In the process of impoverishing a face, boundary vertices cannot be removed but surrounded vertices can be removed.
- The `world`, `eye`, and `screen` fields store the geometrical information of a vertex as three cartesian coordinates. The *world* and *eye* are *xyz*-coordinates normalised. While the *eye* is changed by the orientation of a face, the *world* remain unchanged in the whole process. The *screen* is the screen coordinates obtained from the *eye*.
- The `normal` is an average of the surface normals of polygons shared by the vertex, which is based on the *eye* field. It is usually used for creating an image and impoverishing a face.
- The `pl` is a pointer to a list of polygons sharing the vertex.
- Finally, the `next` is a pointer to another vertex.

2.2.2 Edge

An edge is a structure consisting of four fields: `v1`, `v2`, `pl`, and `next` field. The *v1* and *v2* are a pair of pointers to two vertices belonging to an edge. The *pl* is a list of polygons sharing an edge. The *next* is a pointer to another edge.

On the other hand, an edge is divided into two types according to its attributes. One is a boundary type and the other is a shared type. A boundary

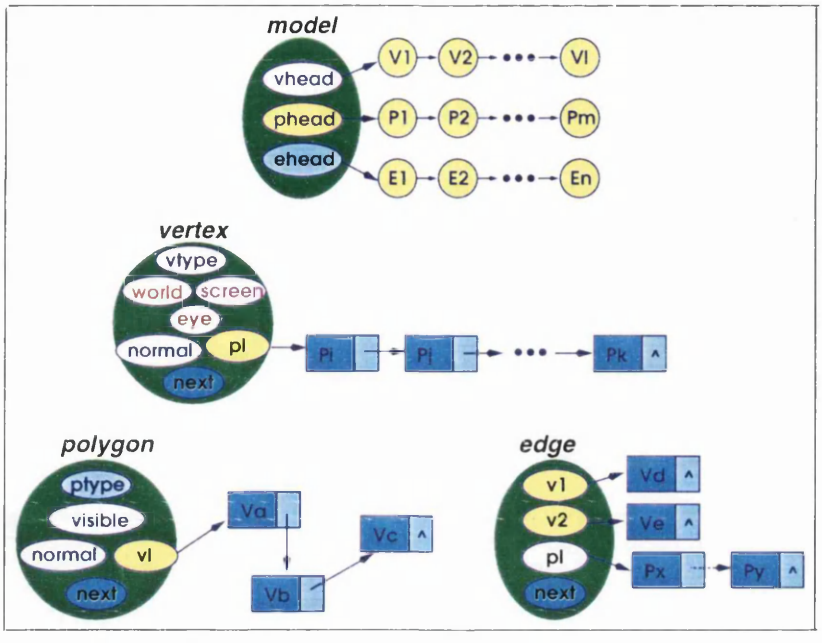


Figure 3.4: Data structure for a face model

edge contains a single polygon in its polygon list, that is, the vertices belonging to it are both boundary ones, while a shared edge contains two polygons. In editing a face using the *Meditor*, the boundary edge cannot be swapped but the shared one can be swapped.

2.2.3 Polygon

A polygon is a structure consisting of five fields: `ptype`, `vl`, `normal`, `visible`, and `next`. The *ptype* contains the information that a polygon shares a vertex which is either to be removed or not when impoverishing a face. The *vl* is a pointer to a list of three vertices forming a triangle[3]. The vertices of the triangle are ordered anti-clockwise.

The *normal* is a surface normal perpendicular to the triangle, which is obtained from the eye-coordinates of vertices linked to the *vl*. The *visible* field gives the information that a polygon is visible in the eye direction. Finally, the *next* points to another polygon.

[3]In this sense, a polygon is often referred to as a triangle, as well.

2.3 Vertex geometry

Vertices in a face model can be divided into three families: adjacent, surrounded, and boundary vertices. An *adjacent* vertex is a neighbouring vertex connected by an edge. A *surrounded* vertex is surrounded by a set of triangles and has a closed path linking up adjacent vertices to form a *ring*. If a vertex is not surrounded by triangles, it is a *boundary* vertex.

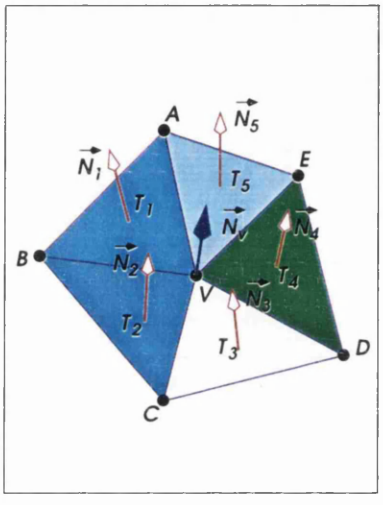


Figure 3.5: An example of vertex geometry

This picture partially shows the geometry of a vertex of a face. The vertex V is called a *surrounded* vertex. The vertices A, B, C, D, and E are called *boundary* vertices, which are adjacent to the *surrounded* vertex V. These adjacent vertices make a closed path, called a *ring*. The *surrounded* vertex V is removed by comparing surface normals $\vec{N}_i (i = 1, 2, ..., 5)$ with the vertex normal $\vec{N}_v$.

Figure 3.5 shows an example of the geometry of a surrounded vertex in a face. The surrounded vertex V has a ring consisting of adjacent vertices A, B, C, D, and E. These adjacent vertices are also boundary vertices, because they have no rings. For example, the vertex A has adjacent vertices B, V, and E. These do not form a ring.

A *surface normal* $\vec{N}_i$ is a normalised vector orthogonal to a triangle, which is defined as:

$$\vec{N}_i(x_i,\ y_i,\ z_i) \ = \ \vec{a} \times \vec{b},$$

where $\times$ represents the cross product of vectors and $\vec{a}$ and $\vec{b}$ are the sides of the triangle as vectors sharing a tail. For a triangle T_1 in Figure 3.5, assuming that $\vec{BV}$ and $\vec{BA}$ are side-vectors, $\vec{N}_1 \ = \ \vec{BV} \times \vec{BA}$.

A *vertex normal* $\vec{N}_w$ is an average of surface normals of triangles shared by the vertex W, which is calculated as follows:

$$\vec{N}_w(x_w,\ y_w,\ z_w) \ = \ \frac{1}{n}(\sum^{n} x_i,\ \sum^{n} y_i,\ \sum^{n} z_i),$$

where $\sum^{n}$ is the component summation of surface normals of all triangles sharing the vertex W, and n is the number of the surface normals. For example, as shown

in Figure 3.5, if the triangles T_1, T_2, T_3, T_4, and T_5 have the surface normal vectors $N_1(x_1, y_1, z_1)$, $N_2(x_2, y_2, z_2)$, $N_3(x_3, y_3, z_3)$, $N_4(x_4, y_4, z_4)$, and $N_5(x_5, y_5, z_5)$ respectively, then the vertex normal N_v is

$$N_v(x_v,\ y_v,\ z_v) \ = \ \frac{1}{5}(\sum^{5} x_i,\ \sum^{5} y_i,\ \sum^{5} z_i).$$

A *critical angle* θ_i formed by both the vertex normal $\vec{N_v}(x_v,\ y_v,\ z_v)$ of a surrounded vertex V and a surface normal $\vec{N_i}(x_i,\ y_i,\ z_i)$ is defined by the dot product as follows:

$$\begin{aligned} \theta_i &= \cos^{-1}(\frac{\vec{N_v} \bullet \vec{N_i}}{|\vec{N_v}| \cdot |\vec{N_i}|}), \\ &= \cos^{-1}(\vec{N_v} \bullet \vec{N_i}), \\ &= \cos^{-1}(x_v x_i \ + \ y_v y_i \ + \ z_v z_i), \end{aligned}$$

where $\bullet$ represents the dot product of two vectors, and $|\vec{N_v}| \cdot |\vec{N_i}| \ = \ 1$ if the magnitudes of the vectors are equal to one.

Similarly, a *neighbouring angle* ϕ_{ij} formed by two surface normals $\vec{N_i}(x_i,\ y_i,\ z_i)$ and $\vec{N_j}(x_j,\ y_j,\ z_j)$ of neighbouring triangles is defined as:

$$\begin{aligned} \phi_{ij} &= \cos^{-1}(\frac{\vec{N_i} \bullet \vec{N_j}}{|\vec{N_i}| \cdot |\vec{N_j}|}), \\ &= \cos^{-1}(\vec{N_i} \bullet \vec{N_j}), \\ &= \cos^{-1}(x_i x_j \ + \ y_i y_j \ + \ z_i z_j), \end{aligned}$$

where $\bullet$ represents the dot product of two vectors, and $|\vec{N_i}| \cdot |\vec{N_j}| \ = \ 1$ if their magnitudes are equal to one.

On the other hand, a surrounded vertex is further classified by three types according to its local curvature. They are convex, concave, and saddle type. These types will mainly be used later in retriangulating a ring. If all the heights of adjacent vertices belonging to a ring are lower than that of a surrounded vertex, it is called a *convex* vertex. On the contrary, if higher, it is called a *concave* vertex. If some of them are lower and others are higher, it is called a *saddle* vertex.

2.4 Ambiguity in retriangulation

In the process of retriangulating a ring, we could encounter a quadrilateral which gives an ambiguity. This ambiguous quadrilateral comes into existence when the

vertices are not placed on one plane; for example, when the heights of vertices facing each other in the diagonal direction are identical and the heights of adjacent vertices are not equal, or when the heights of all vertices are different from one another.

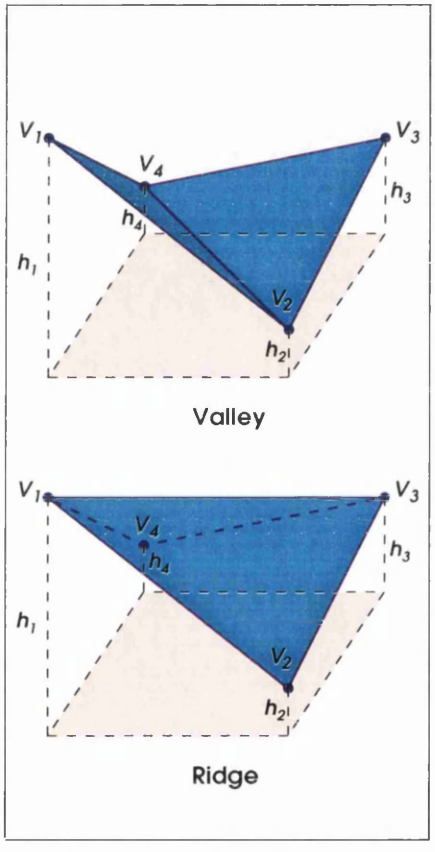


Figure 3.6: An ambiguous quadrilateral

In the process of retriangulation, the ambiguous quadrilateral comes into existence when the vertices are not placed on one plane. The heights of vertices are different from each other. There are two ways of triangulation. One includes the `valley` V_2V_4 as shown on the upper row, which is employed by the lowest method, and the other includes the `ridge` V_1V_3 as shown on the lower row, which is employed by the highest method. We have four possible triangles $\triangle V_1V_2V_4$, $\triangle V_2V_3V_4$, $\triangle V_1V_3V_4$, and $\triangle V_1V_2V_3$.

Let us consider the case, shown in Figure 3.3. We have two reciprocal ways to retriangulate this ambiguous quadrilateral. The two triangles produced by each approach share an edge as either a *valley* or a *ridge*, as shown in Figure 3.6. The edge V_2V_4 represents the valley dividing the quadrilateral into two triangles $\triangle V_1V_2V_4$ and $\triangle V_2V_3V_4$. On the other hand, the edge V_1V_3 shows the ridge dividing it into triangles $\triangle V_1V_3V_4$ and $\triangle V_1V_2V_3$.

The choice of either a ridge or a valley should be determined by the local surface-curvature around the quadrilateral. There is one easy way to validate this choice. If all the vertex heights of the quadrilateral are higher than the height of a removed (surrounded) vertex, then we choose the valley. If they are lower, then we choose the ridge. If we cannot choose either them, the retriangulation is achieved by collecting the neighbouring vertices.

3 Meducer : a mesh reducing tool

Mesh simplification refers to the problem of reducing the number of vertices or polygons in a dense mesh, and yet preserving the overall topology of the mesh. There has been a recent explosion of publications in this field. Much of this work focuses on reconstructing 3D objects from data captured by a laser scanner, or Computed Tomography (CT) [RO96, AS96, RR96, OP96, CCMS97].

For example, a typical model of a human face produced by CT may contain over one million polygons [KT96]. In computer graphics, it is insufficient to use such a dense face mesh in order to display a distant, point-like, face on the screen. Reducing the complexity of a mesh is therefore a must for the graphics system's performance.

Although there are a number of algorithms[4] for mesh simplification, [HL88, Tur92, SZL92, HDD+93, CS94, KT96, CCMS97], it is difficult to adapt them for face impoverishment. There are several reasons. Firstly, they are too dedicated to their applications; that is, there is no one general purpose algorithm that can simplify any given mesh. For example, a successful method for a building-like model may not be applicable to a face. Secondly, it is difficult to decide which one is in a class of its own for our purpose. Thirdly, the data structure used for their models is so different from ours that it is too complicated to adopt their algorithms [EM94]. Finally, it is not difficult to develop a new algorithm for simplifying our faces, because the faces used in this thesis are simple comparing with a complex machine model or a complete human head model. We, therefore, developed the *Meducer*[5] system for impoverishing a face model. This is based on geometric vertex reduction.

3.1 Main stream of the Meducer

The *Meducer* impoverishes a face through four basic tasks: comparing normals, removing a surrounded vertex, producing a ring, and then retriangulating the ring locally. Figure 3.7 shows the main links between these four tasks.

The surrounded vertex V in Figure 3.5, for example, can be removed by the comparison of normal vectors. That is, if all the critical angles θ_is between $\vec{N}_v$ and $\vec{N}_i$ $(i = 1, ..., 5)$ are less than a prespecified angle ε, the vertex V is removed. It leaves a ring consisting of adjacent vertices, such as $ABCDE$. This ring is retriangulated in a way that approximates the previous surface as optimally as

[4] *Appendix A* collectively introduces several algoritms of them.

[5] This word stands for MEsh reDUCER.

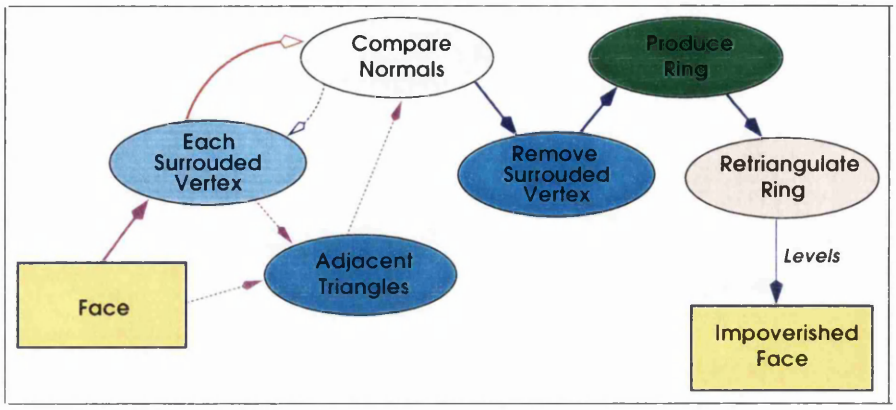


Figure 3.7: Main stream of the *Meducer*

possible. We will introduce three operations to retriangulate the ring $ABCDE$ separately.

The impoverished levels in Figure 3.1 show an example by the *Meducer* program. They are increasingly difficult to recognise as the impoverishment goes further, although they preserve both the global and local shape of the original face. For example, the 80% impoverished face preserves its nose, however it is recognizable more as a `potato-shaped` face rather than as the original one. It should be noticed that the potato-shaped face contains discontinuous variations in intensity. They are attributed to radical changes of surface orientations between adjacent triangles, or to very thin or small triangles. To correct these undesired triangles, we have developed a mesh editor (*Meditor*), which will be discussed later in this chapter.

3.2 Retriangulating a ring

The removal of a surrounded vertex having a very low curvature produces a ring which is a polygon consisting of adjacent vertices. The ring is then retriangulated so that it preserves the previous topology as much as possible. In this thesis, the retriangulation of a ring is achieved by one of three operations: the high, low, and neighbourhood operations.

The choice of one operation depends on three types of the local curvature in a surrounded vertex. If it is a convex vertex, the high operation is selected. If it is a concave vertex, the low operation is selected. Otherwise, the neighbourhood operation is selected.

Figure 3.8 shows an example of impoverished faces created by enforcing the application of three retriangulation operations, independently. The leftmost face

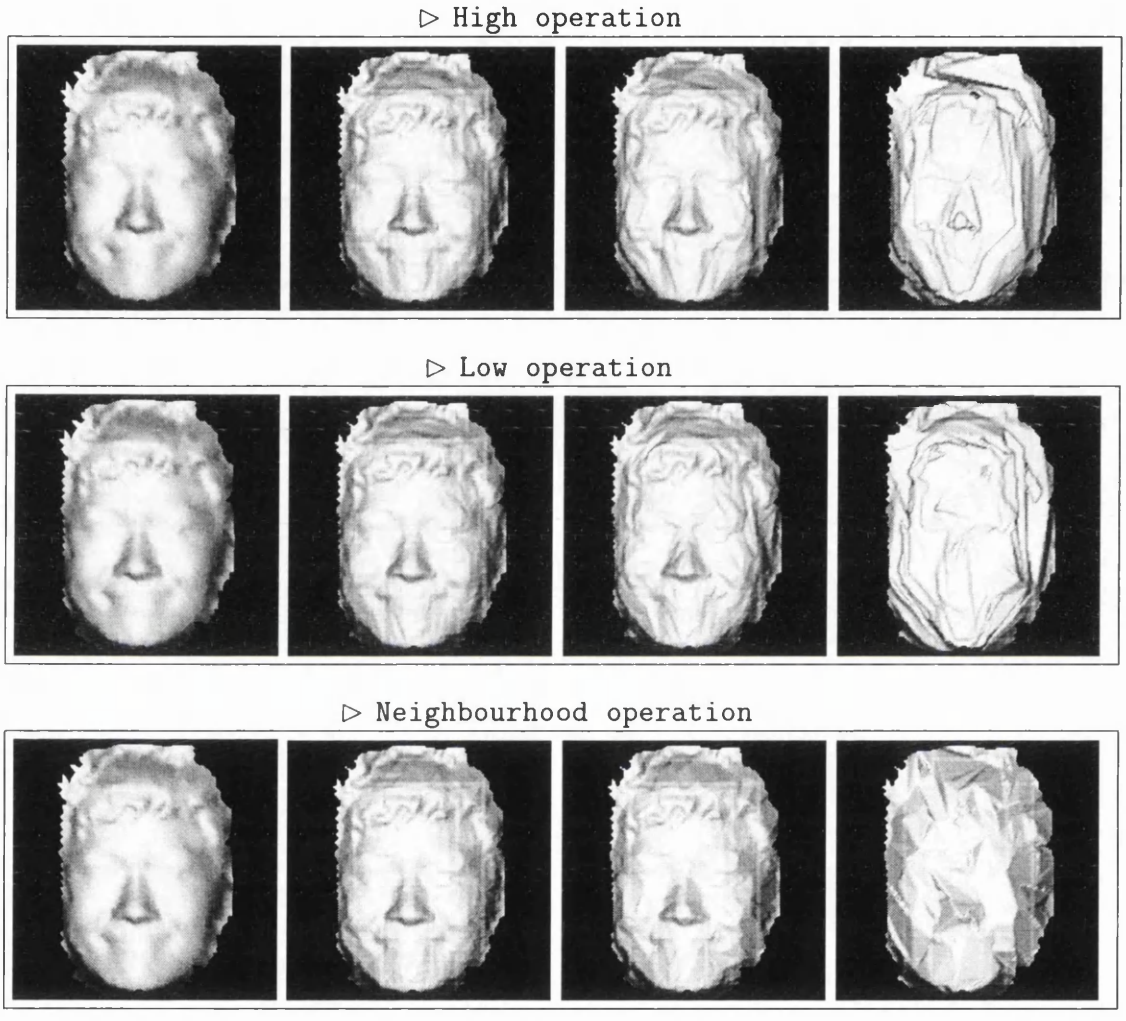


Figure 3.8: An example of retriangulation operations

Each of retriangulation operations is forcibly applied independently to the original faces on the leftmost columns, which are the same as one in Figure 3.1. They consist of 3339 triangles. Each column shows the same impoverishment levels.

is the original one in each row and the rest show the impoverished faces, which are impoverished by 20%, 40%, and 80% from left to right respectively. Each operation preserves the global topology of the original faces.

3.2.1 High operation

Let us consider a new view of Figure 3.5 in the direction of x-axis as shown in Figure 3.9. The height H_v of the surrounded vertex V on yz-plane is higher than those of the adjacent vertices A, B, C, D, and E in a ring $ABCDE$, so the vertex V is a convex vertex. In this case the `high` operation is applied.

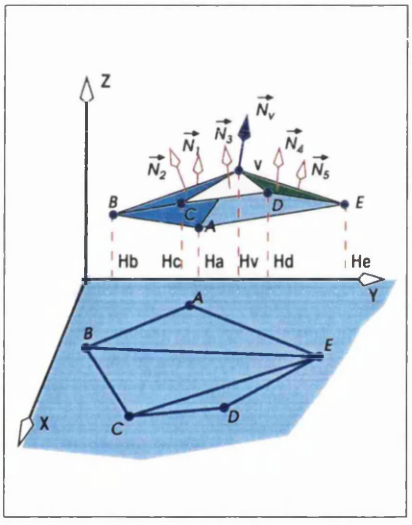


Figure 3.9: High operation

When the surrounded vertex V is a convex vertex, the `high` operation is applied. The ring $ABCDE$ is retriangulated locally in a way that the three highest vertices create a new triangle. We firstly obtain one triangle $\triangle DEC$, and then $\triangle EBC$, and finally $\triangle EAB$, provided that a descending order of vertex heights is H_d, H_e, H_c, H_b, and H_a. The xy-plane shows three triangles after retriangulation.

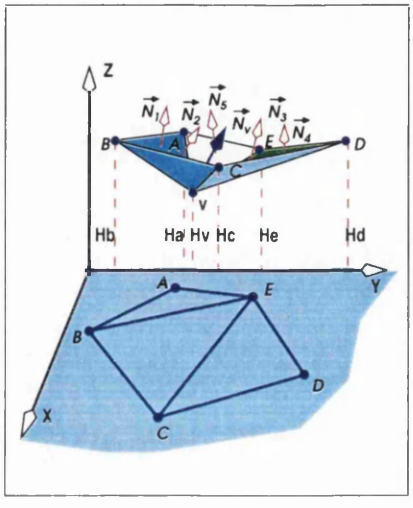
The ring $ABCDE$ is retriangulated locally in a way that a new triangle is made of three highest vertices selected from the ring. Provided that a descending order of the heights corresponding to the adjacent vertices is H_d, H_e, H_c, H_b, and H_a, we first obtain one triangle $\triangle DEC$, and then the other $\triangle EBC$, and finally another $\triangle EAB$. The xy-plane in Figure 3.9 shows the three triangles after the retriangulation through this operation.

3.2.2 Low operation

Let us consider another view of Figure 3.5 in the direction of x-axis as shown in Figure 3.10. The height H_v of the surrounded vertex V on yz-plane is now lower than those of adjacent vertices A, B, C, D, and E in a ring $ABCDE$, so the vertex V is a concave vertex. In this case the `low` operation is applied rather than the others.

Figure 3.10: Low operation

When the surrounded vertex V is a concave vertex, the `low` operation is applied. The ring $ABCDE$ is retriangulated in a way that the three lowest vertices make a new triangle. We firstly obtain one triangle $\triangle CEB$, and then $\triangle CDE$ and finally $\triangle ABE$, provided that an ascending order of heights is H_c, H_e, H_b, H_d, and H_a. The xy-plane shows three triangles after retriangulation.



This operation allows the ring $ABCDE$ to be retriangulated in a way that three lowest vertices form a triangle. Accordingly, we first obtain one triangle $\triangle CEB$, and then $\triangle CDE$ and finally $\triangle ABE$, provided that an ascending order of their heights is H_c, H_e, H_b, H_d, and H_a. The xy-plane in Figure 3.10 shows three triangles after performing the retriangulation.

3.2.3 Neighbourhood operation

Let us consider again a third view of Figure 3.5 in the direction of x-axis as shown in Figure 3.11. The height of the surrounded vertex V is neither lower nor higher than those of adjacent vertices A, B, C, D, and E. If the vertices B and E are higher than V, and the rest of them are lower than that, so the vertex V is a saddle vertex. In this case, the `neighbourhood` operation is applied.

As shown in Figure 3.5, the triangle T_1 and T_2 are neighbouring, which have a shared edge VB, but T_1 and T_3 are not. Retriangulation can be accomplished by merging these neighbouring triangles according to neighbouring angles ϕ_{ij} between their surface normals $\vec{N}_i$ and $\vec{N}_j$, where i and j are $1, 2, ..,$ or 5, and $i \neq j$.

One possible way is to merge the adjacent triangles forming a minimum neighbouring angle. Provided that an ascending order of the neighbouring angles is ϕ_{12}, ϕ_{45}, ϕ_{34}, ϕ_{23}, and finally ϕ_{15}, we have new triangles as shown on xy-plane in Figure 3.11. The triangle T_1 and T_2 are so firstly merged that they give a new triangle $\triangle ABC$, and then the triangle T_4 and T_5 give a new triangle $\triangle ADE$, and finally the rest of triangles give a triangle $\triangle ACD$ as well.

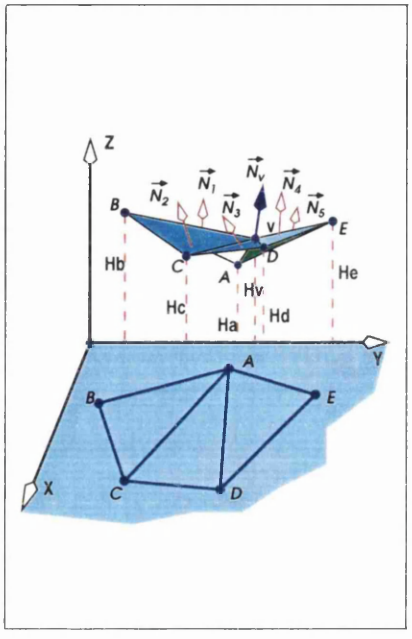


Figure 3.11: Neighbourhood operation

When the surrounded vertex V is a saddle vertex, the `neighbourhood` operation is applied. The ring $ABCDE$ is retriangulated in a way that the neighbouring triangles having a minimum angle between them are merged. We merge firstly triangles T_1 and T_2, and then T_4 and T_5, provided that an ascending order of the neighbouring angles is ϕ_{12}, ϕ_{45}, ϕ_{34}, ϕ_{23}, and ϕ_{15}. The xy-plane shows three triangles after retriangulation.

3.2.4 Contributions of each operation

In general, in the process of impoverishing a face through the *Meducer*, we found that the retriangulation is mostly performed by the neighbourhood operation. Figure 3.12 shows each contribution trend of three operations with respect to the impoverished levels. For example, the 20% impoverished face in Figure 3.1 is retriangulated by the neighbourhood operation over 90% of the total number of operations, while the high and low operations are used in less than 10% of the process. This is an evidence that the original face is mostly smooth rather than jagged.

4 Meditor : a mesh editing tool

As mentioned so far, the *Meducer* works well without changing the global topology of a face as it impoverishes the face. However, it suffers from a problem of undesired triangles. These undesired triangles are produced in a highly impoverished face as shown on the left-hand side in Figure 3.13, which is 80% impoverished.

The undesired triangles are very thin, tiny, or even overturned, which look like *Mach* bands in the image. They may cause a reconstructed face to diverge from the original face. To correct those undesired triangles in an impoverished

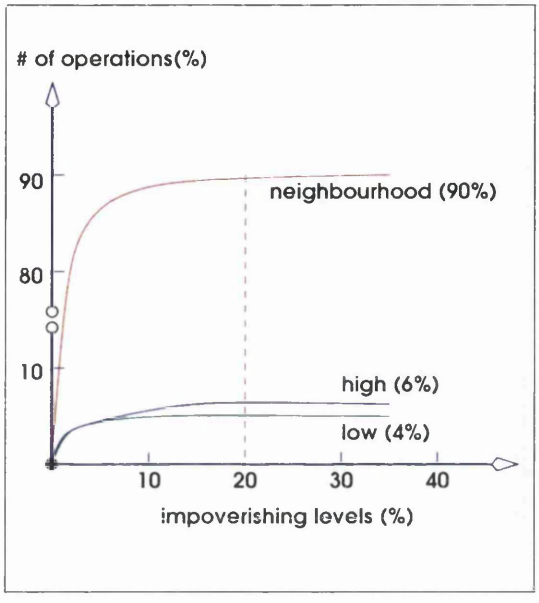


Figure 3.12: An example of contributions of three retriangulation operations

For the original face shown in Figure 3.1, the neighbourhood operation mainly contributes to retriangulate it over 90% of the total number of operations. On the other hand, the high and low operations are used less than 10%.

face, we have developed the *Meditor*[6], which is a tool for editing a face model interactively.

The right-hand side in Figure 3.13 shows a face that was edited using the *Meditor*. The undesired triangles are mostly corrected. The edited face is 90% impoverished, and yet it is more recognizable than the 80% impoverished face.

The *Meditor* is endowed with five basic operations that are specified by the user. These operations follow one of three searches for a geometrical element, such as a vertex, edge, or triangle, in a face. The searches are also invoked by a Mouse Action (MA)[7]. For example, if we are going to remove a vertex in the face interactively, it can be selected by clicking the mouse on the screen, which requires a search for that vertex in the data structure of the face. Figure 3.14 shows the classification of searches and operations invoked by a *MA*.

4.1 Searches

Meditor offers three types of search. One is to search for a vertex in a face, another is for an edge, and a third is for a triangle. In practice, these searches are performed in 2D space, because a *MA* yields only a 2D point information on the screen. We assume that the *MA* returns a point $Q(x_q,\ y_q)$.

[6]This word stands for MEsh eDITOR.

[7]In this section, it means the action of clicking the mouse on the screen by the user.

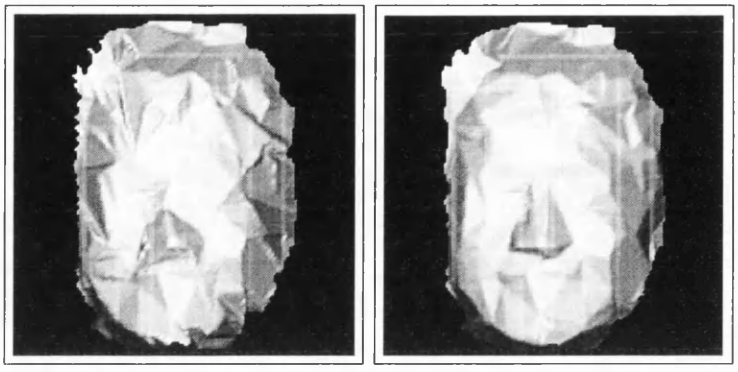


Figure 3.13: Mesh editing with Meditor

An 80% impoverished face produced by our own impoverishing method (left) has the undesired triangles which look dark and scratched. After applying the meditor, the undesired triangles are removed and the level of impoverishment becomes 90% (right).

4.1.1 Vertex search

For a point $Q(x_q,\ y_q)$ corresponding to a MA, a selected vertex is closest to it. The closest vertex may be found by comparing the distance between Q and a vertex $V(x_v,\ y_v)$ of a face.

In general, the distance between two points $Q(x_q,\ y_q)$ and $V(x_v,\ y_v)$ is defined as follows:

$$|QV| = \sqrt{(x_v - x_q)^2 + (y_v - y_q)^2}.$$

4.1.2 Edge search

For a point $Q(x_q,\ y_q)$ corresponding to a MA, a desired edge is closest to it. The closest edge may be detected by comparing the shortest distance between $Q(x_q,\ y_q)$ and an edge AB of a face, which consists of two vertices A and B.

4.1.2.1 Distance between a point and an edge Figure 3.15 shows a representation of vectors according to an edge AB and a point $Q(x_q,\ y_q)$ corresponding to the MA. Let a point on the edge AB be H, which satisfies that $\vec{QH}$ is perpendicular to $\vec{AB}$.

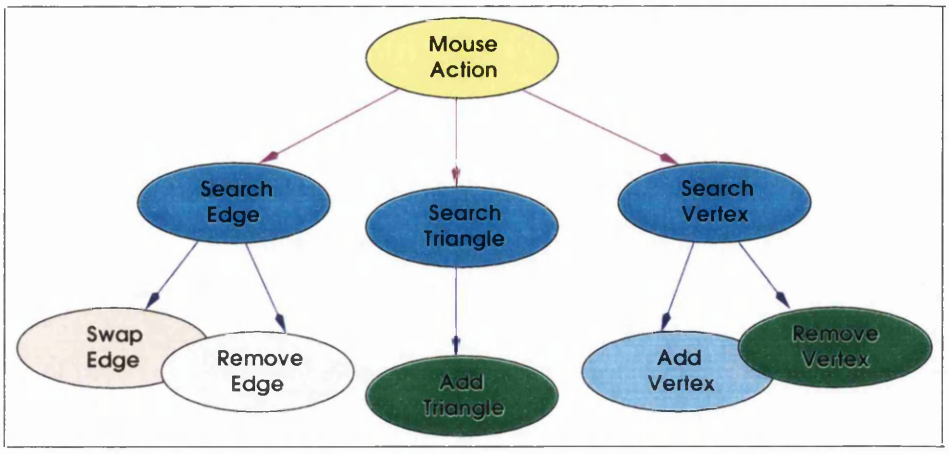


Figure 3.14: Searches and operations of the *Meditor*

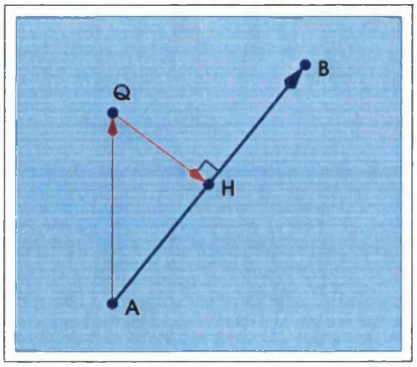


Figure 3.15: $|QH|$ is the shortest distance from a point Q to an edge AB

A line equation α passing through two vertices $A(x_a,\ y_a)$ and $B(x_b,\ y_b)$ may be defined as

$$\alpha : \quad \frac{x - x_a}{x_b - x_a} = \frac{y - y_a}{y_b - y_a} = t,$$

where $P(x,\ y)$ is an arbitrary point on the line α and t is a real number.

Since the point H is on the line α, the xy-coordinates of it are

$$H(x_a + t(x_b - x_a),\ y_a + t(y_b - y_a)).$$

Now if we obtain the value of t, the shortest distance between $Q(x_q,\ y_q)$ and AB; that is, $|QH|$ can be computed easily.

Since $\vec{QH}$ is perpendicular to $\vec{AB}$, the value of t is obtained as follows:

$$\begin{aligned} \vec{QH} \bullet \vec{AB} &= (x_a - x_q + t(x_b - x_a),\ y_a - y_q + t(y_b - y_a)) \bullet (x_b - x_a,\ y_b - y_a) \\ &= (x_a - x_q + t(x_b - x_a))(x_b - x_a) + (y_a - y_q + t(y_b - y_a))(y_b - y_a) \\ &= 0. \end{aligned}$$

Therefore,

$$t = \frac{(x_q - x_a)(x_b - x_a) + (y_q - y_a)(y_b - y_a)}{(x_b - x_a)^2 + (y_b - y_a)^2}$$

$$= \frac{\vec{AQ} \bullet \vec{AB}}{|\vec{AB}|^2}$$

Consequently, using this value of t, we can compute the distance between Q and H as follows:

$$|QH| = \sqrt{(x_a - x_q + t(x_b - x_a))^2 + (y_a - y_q + t(y_b - y_a))^2}.$$

4.1.3 Triangle search

The problem of searching a corresponding triangle in a set of triangles may be stated formally as follows:

- For a point $Q(x_q, y_q)$ specified by a MA and a triangle $\triangle ABC$ consisting of three vertices $A(x_a, y_a)$, $B(x_b, y_b)$, and $C(x_c, y_c)$, decide whether the point Q lies on the inside of the triangle $\triangle ABC$.

To reduce the complexity of the solution for this problem, we allow the point Q only to lie on the inside of an inscribed circle on the triangle $\triangle ABC$.

4.1.3.1 An inscribed circle An inscribed circle is the biggest circle which lies inside the triangle as shown in Figure 3.16. Each side of the triangle $\triangle ABC$ is a tangent line touching the circle G.

For example, the side $\overline{BC}$ touches it at a point H, and the line segment $\overline{GH}$ is perpendicular to the side $\overline{BC}$, which is the radius R of the circle G. The line segments from the incentre G to each vertex A, B, and C bisect each vertex angle. For example, if the vertex angle $\angle ABC$ is equal to 2β, then both $\angle ABG$ and $\angle GBC$ are identical with β.

Now if we can compute the xy coordinates for the centre of G and its radius, R, it is easy to examine whether the point Q specified by the MA lies on the inside of the circle G.

Provided that the position vectors corresponding to the vertices of the triangle $\triangle ABC$ are $\vec{a}$, $\vec{b}$, and $\vec{c}$ respectively, the radius R and the centre G are

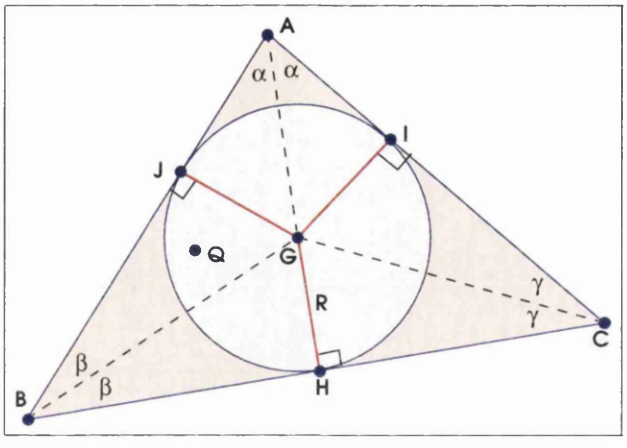


Figure 3.16: An inscribed circle G on a triangle $\triangle ABC$

defined as follows [Gla90, Kir92]:

$$R = \frac{|\vec{a} \times \vec{b} + \vec{b} \times \vec{c} + \vec{c} \times \vec{a}|}{|\vec{a} - \vec{b}| + |\vec{b} - \vec{c}| + |\vec{c} - \vec{a}|},$$

$$G = \frac{|\vec{b} - \vec{c}|\vec{a} + |\vec{c} - \vec{a}|\vec{b} + |\vec{a} - \vec{b}|\vec{c}}{|\vec{a} - \vec{b}| + |\vec{b} - \vec{c}| + |\vec{c} - \vec{a}|},$$

where $\times$ represents the cross product and vertical bars denote the magnitude of the vector.

4.1.3.2 An inside point We now have an inscribed circle of a triangle $\triangle ABC$, which is centred at $G(x_g, y_g)$ and has a radius R. Therefore, the equation of the circle G is defined as follows:

$$(x - x_g)^2 + (y - y_g)^2 = R^2,$$

where $P(x, y)$ is an arbitrary point on the circle G. If the point $Q(x_q, y_q)$ is placed the inside of the circle, the distance between G and Q must be smaller than the radius R. Therefore, the inside point Q satisfies as follows:

$$(x_q - x_g)^2 + (y_q - y_g)^2 < R^2.$$

4.2 Operations

There are five basic operations for editing an impoverished face: vertex removal, vertex addition, triangle addition, edge swap, and edge removal.

4.2.1 Vertex removal

Vertex removal is the act of deleting a vertex in a face, which is detected by a vertex search. In fact, it deletes a vertex corresponding to a *MA* from a list of vertices (*vhead*), to delete polygons sharing the vertex from a list of polygons (*phead*), and to delete edges that share the vertex from a list of edges (*ehead*).

For example, in Figure 3.17, the left-hand side shows a face model before removing the selected vertex *A*, while the right-hand side shows the model after removing. This operation is useful to remove a vertex which shares the undesired triangles.

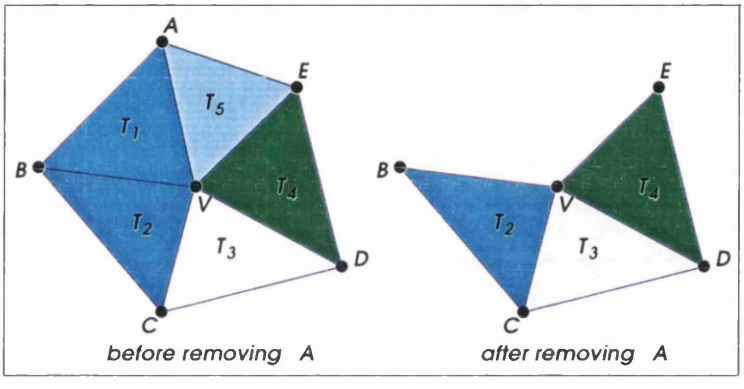


Figure 3.17: An example of vertex removal operation

4.2.2 Vertex addition

Vertex addition is the act of adding a vertex inside a triangle activated by a *MA*. This requires a triangle search. In fact, it adds a vertex into a list of vertices (*vhead*), to create three new triangles, which share the added vertex, and to add them in a list of polygons (*phead*).

An added vertex is the centre of gravity G for a specified triangle $\triangle ABC$ by the *MA*. The xyz-coordinate components of the gravity centre $G(x_g,\ y_g,\ z_g)$ are obtained as follows:

$$G(x_g,\ y_g,\ z_g) \ = \ \frac{1}{3}(x_a + x_b + x_c,\ y_a + y_b + y_c,\ z_a + z_b + z_c),$$

where $A(x_a,\ y_a,\ z_a)$ $B(x_b,\ y_b,\ z_b)$, and $C(x_c,\ y_c,\ z_c)$ are vertex coordinates.

Figure 3.18 shows an example of vertex addition operation; the left-hand side is a triangle as part of a face before addition, while the other one is three triangles after the addition of the vertex G. This operation is useful to retain specific vertices; for example, the heighest or lowest vertex on the nose.

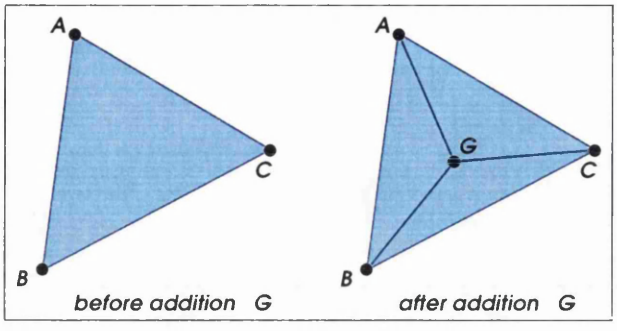


Figure 3.18: An example of vertex addition operation

4.2.3 Triangle addition

If there are three vertices specified by three MAs, triangle addition is the act of adding a triangle consisting of those vertices into a face. In practice, this operation creates a triangle consisting of three specified vertices and adds it to a list of polygons (*phead*).

Figure 3.19 shows an example of the triangle addition operation. The left-hand side is a face consisting of several tiny triangles. After removing the vertices A_1, A_2, A_3, A_4, and A_5, a new triangle $\triangle ABV$ was added as shown in the right-hand side. Together with the vertex removal operation, this operation is useful to merge thin or tiny triangles into a bigger triangle.

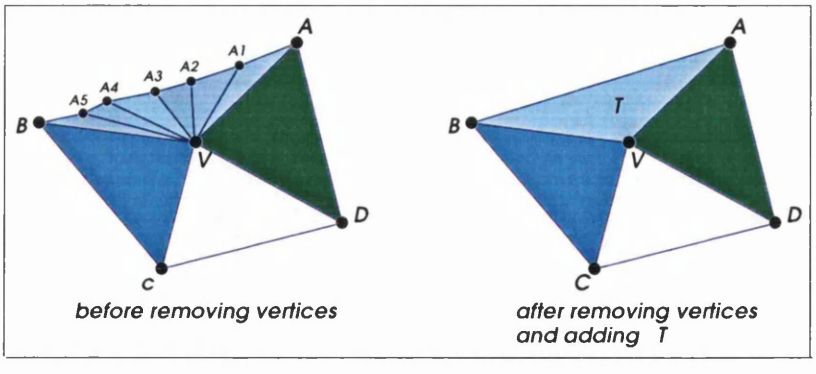


Figure 3.19: An example of triangle addition operation

As similar with the vertex removal operation, this requires a search for three vertices identified by the MAs.

4.2.4 Edge swap

If two triangles share an edge selected by a MA, then edge swapping is the act of removing the two triangles and adding two new triangles sharing a new edge between the vertices that were not on the shared edge. This requires a search of the closest edge to the MA. This removes the selected edge and any triangles sharing it. It then adds a new edge and two new triangles into a list of edges (*ehead*) and polygons (*phead*).

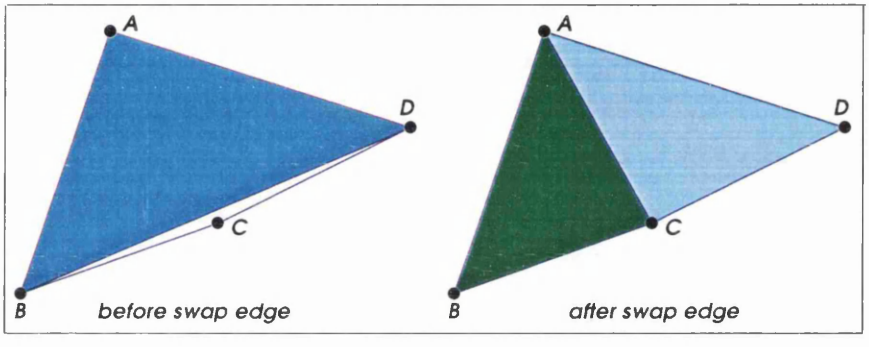


Figure 3.20: An example of edge swap operation

Figure 3.20 shows an example of the edge swap operation. If the edge BD is selected by the MA, two triangles $\triangle ABD$ and $\triangle BCD$ are deleted, while two new triangles $\triangle ABC$ and $\triangle ACD$ are added, which share a new edge AC. It removes the very thin triangle $\triangle BCD$ consequently.

4.2.5 Edge removal

Edge removal is the act of deleting an edge in a face. The edge can be selected by an edge search. This deletes the selected edge from a list of edges (*ehead*) and deletes any triangles sharing it from a list of polygons (*phead*).

For example, in Figure 3.21, the left-hand side shows a face model before removing edges. The middle figure shows the model after removing the boundary edge CD. In this case, the triangle T_3 is deleted. The right-hand side shows the removal of the internal edge VD. In this case, the triangles T_3 and T_4 are deleted. Moreover, if the vertex D is no longer shared by other triangles, it is also deteted.

This operation is useful when we need to remove undesired triangles one by one.

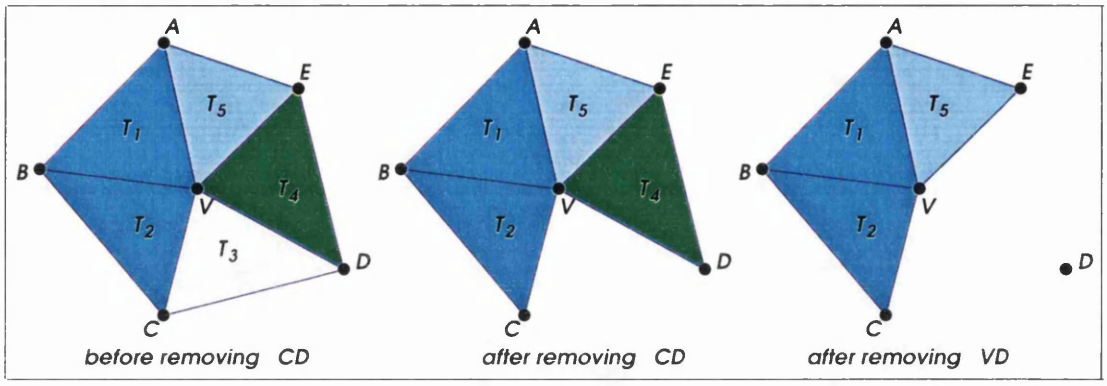


Figure 3.21: An example of edge removal operation

5 Discussion

It should be stressed that the purpose of this chapter was not focused on developing the tools completely but on using them to obtain an impoverished face. All impoverished faces introduced throughout this thesis were created by using these tools.

The *Meducer* tool impoverishes the original face represented by a mesh, which is based on geometric vertex reduction. Firstly it deletes a surrounded vertex which gives a low curvature and then produces a ring which consists of vertices adjacent to the surrounded vertex. Finally, it retriangulates the ring using one of three operations.

The *Meducer* usually works well without changing the global topology of the original face, however, it gives undesired triangles in highly impoverished faces. To remove these triangles, we have developed the *Meditor* which is endowed with five basic operations. One can exploit this tool to polish any remaining problems left with the impoverished faces produced by the *Meducer*.

Chapter 4

Describing Shape from Prior Knowledge

Shape-from-shading (*SFS*) techniques have been intended to solve the shading problem inversely. They are formulated by an image irradiance equation $I = R(p, q)$ as mentioned in *Chapter 2*. Most techniques employ *Lambertian* reflectance and additional constraints in order to obtain a solution (p, q) for every pixel.

A collection of solutions is called a needle diagram [Hor86, WH97], which may give a recovered image lit by a different light source. Most *SFS* techniques proposed so far have explored image regeneration with different light sources, and yet with the same view as an input image.

On the other hand, a depth map of the original surface in an input image can usually be determined by integration along arbitrary curves in a needle diagram, because a depth $z(x, y)$ satisfies that $dz(x, y) = pdx + qdy$ [Hor86]. However a recovered surface is very poor at describing the original surface observed from a different view point, because it is too flat. For example, the face of *Queen Elizabeth II* on a coin cannot describe the different orientation of it. Consequently, most recovered surfaces using the *SFS* methods suffer from flatness. This flatness is due to the over smoothness assumption; that is, it is due to the lack of geometric knowledge of the subject to be reconstructed.

We will overcome the flatness problem by employing knowledge about the geometry of the human face under analysis. Our new method reformulates the conventional shape-from-shading problem into a *shape-from-prior-knowledge* task.

In this chapter, we will discuss how to extract surface normals from an intensity value of an image, using geometric knowledge. In the next chapter we

will reconstruct an approximation of the original face in the image using the extracted normals.

1 Problem Definition

Given an impoverished object and a single image under a point light source, this chapter shows how to extract surface normals corresponding to the original object in the image. In our investigation, this goal will be achieved in terms of human faces and single *Lambertian* images, where no specular component is apparent.

The *SFS* is fundamentally a very difficult mathematical problem. An intensity value determines only an incident angle between a surface normal and a light source vector at a particular point. Accordingly no one can decide a unique normal, even where the light source vector is known in advance. There is an infinite number of normal candidates which form a space cone[1] at the point.

Our new *shape-from-prior-knowledge* method extracts a unique normal from the space cone. This is obtained by a space plane and a sphere. In addition, the extraction of a unique normal is achieved using two space lines and the cone. We can find the geometric components that contribute to the method in *Appendix B*.

2 Extracting a Surface Normal from an Intensity Cone

To extract a surface normal corresponding to a particular image point, we require some assumptions about the image and original surface.

A face image under analysis is formed by the following assumptions as ones usually used in the *SFS*:

- it has a single distant point light source,
- there is a *Lambertian* shading model,
- its orthographic projection has the same z-axis as the viewing direction, and

[1]This cone is referred as an intensity cone and will be introduced in the next section.

❒ a reflectance coefficient (albedo) is a constant value of one.

We also assume that the smoothness constraint about the surface is no longer used, while the geometric surface information is used to overcome the flatness problem in *SFS*. An impoverished face is a collection of geometric information about the prior knowledge of the original face in the image.

2.1 Main stream of extracting a surface normal

Figure 4.1 shows a graph of the main operations for extracting a solution from an infinite number of surface normals. A face image provides an intensity value I at a particular point. The intensity value constructs inversely an intensity cone depending on a source vector $\vec{L}$.

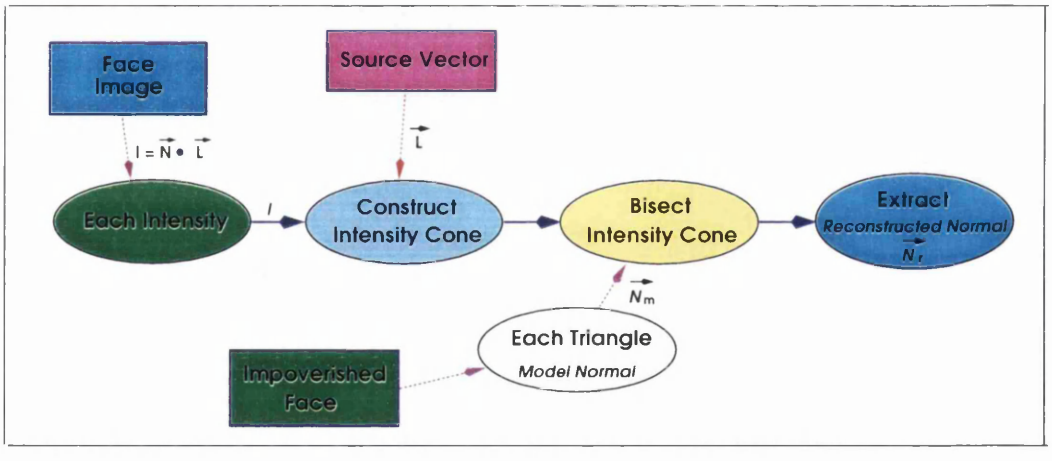


Figure 4.1: Main flow diagram showing the extraction of a surface normal

An intensity cone is a collection of candidates of surface normals. It can be cut into two half cones, by a plane containing a model normal of an impoverished face. Namely, an infinite number of candidates is reduced to only two. The two candidates are on the cutting plane. Consequently, a reconstructed normal is the one that is close to a model normal.

In the next two subsections, we will discuss the intensity cone, and then show about how one candidate is extracted out of an infinite number of potential candidates.

2.2 Intensity cone

In Figure 4.2, let us assume that the point O is an image point having an intensity value I and that a directional vector $\vec{L}(l_x, l_y, l_z)$ towards a single distant point light source passes through a point $L(l_x, l_y, l_z)$. We will call $\vec{L}$ a `source vector`.

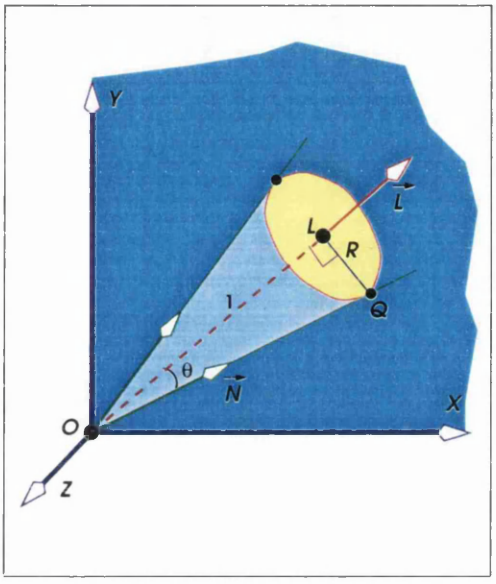


Figure 4.2: An intensity cone

For a given image point O with an intensity value I and a source vector $\vec{L}$, the *Lambertian* reflectance model gives an intensity cone with a constant spread angle θ, a spread radius R, and an axis $\vec{L}$. The candidates of a surface normal $\vec{N}$ lie on the base circle L.

As mentioned in the *Chapter 2*, the *Lambertian* reflectance model $I = \vec{N} \bullet \vec{L}$ gives an intensity value I as a cosine of an angle θ between a surface normal $\vec{N}$ and a source vector $\vec{L}$ at a particular surface point. Inversely, we can obtain a surface normal $\vec{N}$ from this *Lambertian* model provided that an intensity value I and a source vector $\vec{L}$ are given at an image point. However $\vec{N}$ is not unique. There is an infinite number of candidates for $\vec{N}$ satisfying the *Lambertian* model in inverse calculation. They construct a cone in space as shown in Figure 4.2.

Let us discuss how this cone can be obtained inversely from the *Lambertian* reflectance model, that is, $I = \vec{N} \bullet \vec{L} = |\vec{N}| \cdot |\vec{L}| \cos\theta$. We can imagine a space cone formed by $\vec{N}$ when I and $\vec{L}$ are given, because possible vectors $\vec{N}$ making an angle θ with $\vec{L}$ are innumerable round the source vector $\vec{L}$.

An angle θ can be regarded as a spread angle of a space cone. An axis is parallel to a source vector $\vec{L}$ and passes through a centre point L. A base circle L has a radius R of $|LQ|$, which is a bisected area between a plane π (eq. (B.1)) and a sphere σ (eq. (B.2)) as mentioned in *Appendix B*.

It should be stressed that all points on a base circle L satisfy the *Lambertian* reflectance model. They are candidates for the surface normal $\vec{N}$ at an image

point O. Again, these candidates form a 3D circular cone with a constant spread angle θ, a spread radius R, and an axis $\vec{L}$. We will call this cone an `intensity cone` for an image point O.

The spread radius R in Figure 4.2 is easily obtained by trigonometry. The radius R can be obtained from the right triangle $\triangle OLQ$, if we have the angle θ and $|OL|$ is identical with 1. For a given intensity value I, the radius R is given as follows:

$$R = \tan\theta.$$

On the other hand, an angle θ can be obtained as follows:

$$\theta = \cos^{-1}(\frac{I}{|\vec{N}| \cdot |\vec{L}|}) = \cos^{-1} I,$$

where the surface normal $\vec{N}$ and the source vector $\vec{L}$ are unit vectors.

In our investigation, the value I of an image intensity is normalised from 0 to 1. Therefore the angle θ varies between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ radians in trigonometry. However negative angles are not considered, because they never occur in practice. This leads to only positive angles 0 to $\frac{\pi}{2}$. Therefore the spread radius R varies between 0 and an infinite value. If R is equal to zero, there is only one candidate $\vec{L}$. Otherwise, there is an infinite number of candidates. One of these candidates can be extracted by using prior knowledge, even if R is infinite.

2.3 Extracting a surface normal

As shown in Figure 4.2, the candidates of a surface normal $\vec{N}$ construct an intensity cone. At this point, we should confirm a question: which candidate best describes the original surface? If we obtain a surface normal that best approximates the orientation of the original surface from those candidates, then our goal will be achieved.

Fortunately we can obtain an approximating surface normal by employing prior knowledge of the original surface, which is a normal vector from an impoverished face. We will call the approximating surface normal a `reconstructed normal` $\vec{N_r}(X_r, X_y, Z_r)$, and a normal vector from an impoverished face a `model normal` $\vec{N_m}(X_m, Y_m, Z_m)$.

Figure 4.3 shows a bisected intensity cone together with a model normal $\vec{N_m}$ and a reconstructed normal $\vec{N_r}$. Obviously, the reconstructed normal $\vec{N_r}$ is the best candidate; it is nearest to the model normal $\vec{N_m}$ in the *Euclidean* distance.

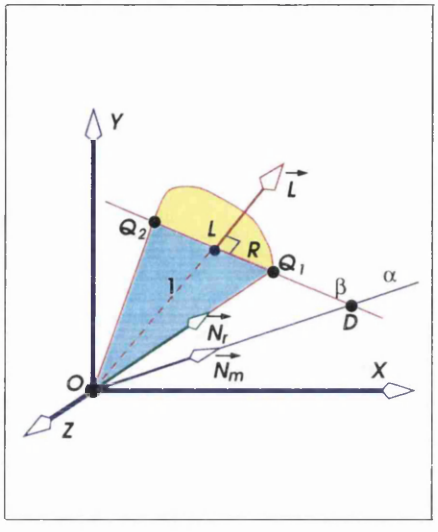


Figure 4.3: A bisected intensity cone

The line α parallel to a model normal $\vec{N}_m$ meets a point $D(a, b, c)$ on the plane π (eq. (B.1) in *Appendix B*) so that the base circle L and the point D are coplanar. The line β passing two points D and L bisects the circle L at two points $Q_1(X_1, Y_1, Z_1)$ and $Q_2(X_2, Y_2, Z_2)$. A reconstructed normal $\vec{N}_r$ is either $O\vec{Q}_1$ or $O\vec{Q}_2$.

It should be stressed that an original surface having an image intensity I can be uniquely described by a reconstructed normal $\vec{N}_r$. So, our goal can be turned to a problem of searching for the nearest candidate $\vec{N}_r$.

A line α parallel to a model normal $\vec{N}_m$ meets at a point $D(a, b, c)$ on the plane π (eq. (B.1) in *Appendix B*) so that the circle L and the point D are coplanar. In addition, a line β, passing through two points D and L, bisects the circle L at two points $Q_1(X_1, Y_1, Z_1)$ and $Q_2(X_2, Y_2, Z_2)$. One of them is nearest to the model normal $\vec{N}_m$.

If the *Euclidean* distance $|DQ_1|$ is shorter than $|DQ_2|$, a reconstructed normal $\vec{N}_r(X_r, Y_r, Z_r)$ is identical with a position vector $O\vec{Q}_1$. If the *Euclidean* distance is longer, then it becomes $O\vec{Q}_2$. The computation details for this reconstructed normal $\vec{N}_r$ can be found in *Appendix C*.

For an intensity value I, until now we have extracted a reconstructed normal $\vec{N}_r$ using prior knowledge in terms of a model normal $\vec{N}_m$. However, we have several exceptional conditions that may prevent us from solving them. For example, if an intensity value I is equal to 0 ($\theta = \frac{\pi}{2}$), then a radius R is infinite ($R = \infty$). In addition, the point $D(a, b, c)$ must exist on the plane π (eq. (B.1)). If $\vec{N}_m$ is perpendicular to $\vec{L}$ ($\vec{N}_m \bullet \vec{L} = 0$), then $D(a, b, c)$ does not exist. As another example, if $\vec{N}_m$ is parallel with $\vec{L}$ ($\vec{N}_m \bullet \vec{L} = \pm 1$), we cannot decide either Q_1 or Q_2. However, these conditions do not prevent us from retaining $\vec{N}_r$. *Appendix C* shows alternatives that avoid these conditions.

3 Discussion

The *SFS* problems are not usually well-posed except in special cases such as a sigular point ($I = 1$) [Oli91a], so they often employ the regularisation constraints. Smoothness is a constraint employed in most *SFS* methods. However they suffer from the flatness of recovered surfaces even from a *Lambertian* image. That is, their solutions compel us to view the result from the same direction as the input image.

To overcome this problem, we employ geometrical prior-knowledge about the face under analysis. Our new method reformulates a conventional *SFS* problem into a *shape-from-prior-knowledge* problem. In this chapter, we have discussed how to extract a reconstructed normal $\vec{N}_r$ at a particular point in a *Lambertian* image using a normal $\vec{N}_m$, taken from the prior knowledge of a face model.

An intensity value I constructs an intensity cone with an axis parallel to a source vector $\vec{L}$ and a spread angle θ. A base circle of this cone consists of innumerable candidates of surface normals, which satisfy the *Lambertian* model. A plane containing a model normal $\vec{N}_m$ and the centre L of the base circle, bisects the intensity cone and cuts an infinite number of candidates down to only two. One of them is a reconstructed normal $\vec{N}_r$.

In the next chapter, we will discuss in detail the processes of reconstructing an approximation of the original face fused into an image, using model and reconstructed normals. In addition, the face reconstruction will be conducted on the $90^{+}\%$ impoverished face, shown in Figure 3.1 in *Chapter 3.*

Chapter 5

Face Reconstruction

In the previous chapter, we have discussed how to extract a reconstructed normal $\vec{N_r}$ from an intensity value I and a model normal $\vec{N_m}$. The reconstructed normal is a unique solution for the orientation of the original surface patch at a particular point in an image.

If we can transform a surface patch having a model normal $\vec{N_m}$ into a new surface patch oriented by the reconstructed normal $\vec{N_r}$, then the original surface patch is identical with the newly-oriented patch. This can be achieved by rotating patches in $3D$ space. In order to rotate a patch, we employ a quaternion multiplication. Our new technique gives a robust means for reconstructing the original face of a single image, starting from a very impoverished face.

Figure 5.1 shows one example of many reconstructions. The upper-left is a synthetic rendering of an image. The upper-right is a poor prior-knowledge face, which is $90^{+}\%$ impoverished. This includes the prior knowledge less than 10% and it is no longer considered as the original face of the input image. However the reconstructed face on the bottom is quite recognisable. In addition, it can be viewed in different directions.

In this chapter, we focus on how to reconstruct a very impoverished face using model and reconstructed normals.

1 Main Stream of Face Reconstruction

Figure 5.2 shows the main flow of face reconstruction operation. A reconstructed normal $\vec{N_r}$ and a model normal $\vec{N_m}$ produce a unit quaternion q, which is represented in terms of a rotation angle and a rotation axis. It rotates a triangle

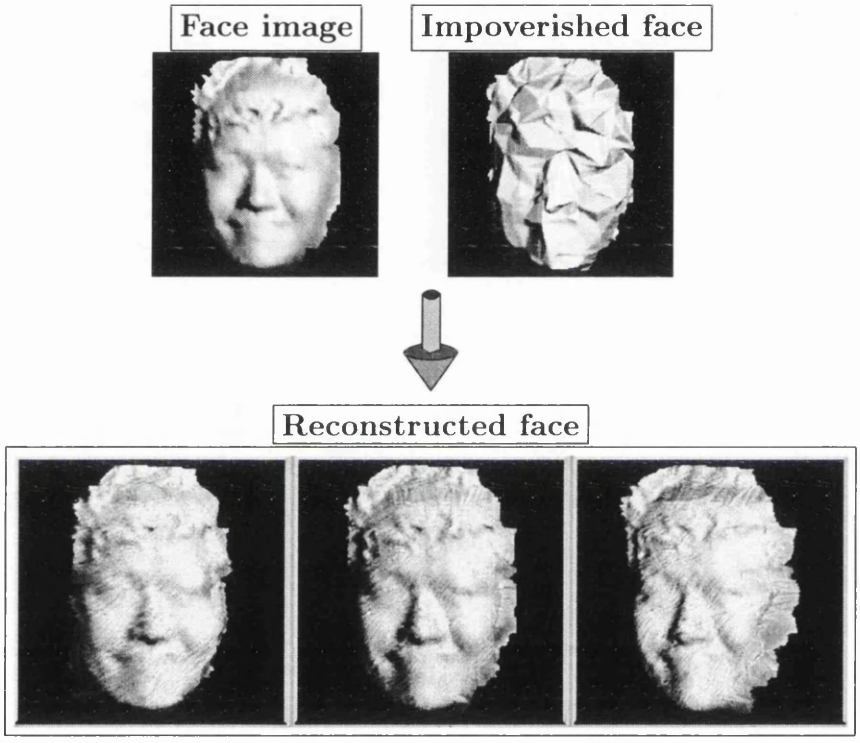


Figure 5.1: An example of reconstruction from a very impoverished face

The upper-left is a synthetic rendering of an image. The upper-right is a poor prior-knowledge face, which is $90^{+}\%$ impoverished. This includes the prior knowledge less than 10% and it is no longer considered as the original face. However the reconstructed face on the bottom is quite recognisable. In addition, it can be viewed in different directions.

specified by the model normal $\vec{N}_m$ so that a new triangle is specified by the reconstructed normal $\vec{N}_r$.

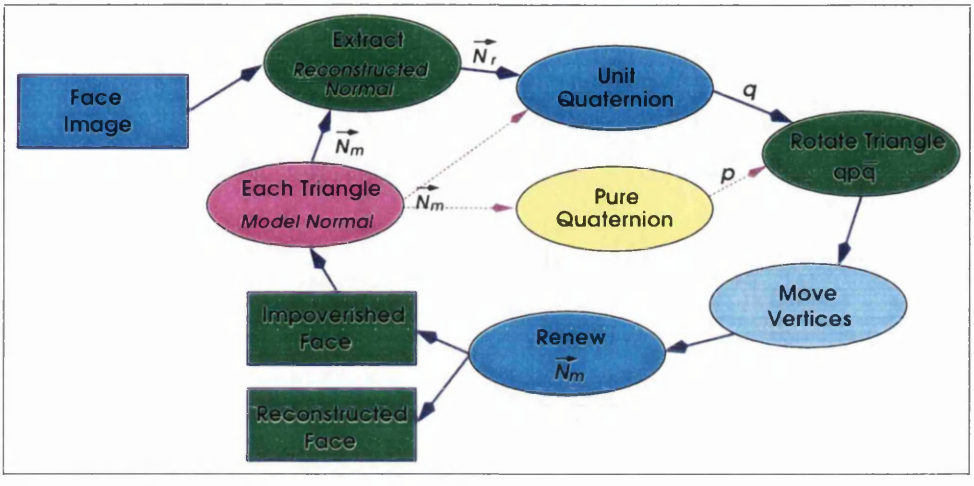


Figure 5.2: Main stream of face reconstruction

The vertices of a rotated triangle are repositioned so as to allow them only to change in depth. Again, they have the same xy-coordinates as before. This moving vertices problem will be discussed in detail later in this chapter.

Now the moved vertices introduce new normal vectors to triangles sharing them. So those triangles must renew their model normals. This procedure is propagated through all triangles of an impoverished face. We call it one `iteration` of the reconstruction procedure.

The next sections will discuss the flow of the face reconstruction operations focusing on quaternions, rotation of a triangle, movement of vertices, and renewal of model normals.

2 Quaternions for rotation

First of all, let us obtain an amount of a `rotation angle` ϕ and a `rotation axis` $\vec{n}(X_n, Y_n, Z_n)$ from both $\vec{N}_r$ and $\vec{N}_m$ as shown in Figure 5.3. The angle ϕ is an angle between $\vec{N}_m(X_m, Y_m, Z_m)$ and $\vec{N}_r(X_r, Y_r, Z_r)$, and the axis $\vec{n}$ is their cross product whose direction is orthogonal to the plane on which they lie.

We can get a rotation angle ϕ using a dot product of two vectors $\vec{N}_m$ and $\vec{N}_r$:

$$\vec{N}_m \bullet \vec{N}_r = |\vec{N}_m| \cdot |\vec{N}_r| \cos\phi$$

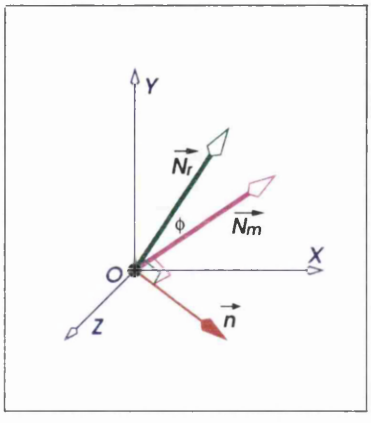


Figure 5.3: A rotation angle and axis

A triangle having $\vec{N_r}$ is obtained by a rotation of a model triangle oriented by $\vec{N_m}$. The rotation can be specified by a rotation angle ϕ and a rotation axis $\vec{n}$ at a particular point O. The angle ϕ is an angle between $\vec{N_m}$ and $\vec{N_r}$, and the axis $\vec{n}$ is a cross product of $\vec{N_m}$ and $\vec{N_r}$.

If these vectors are normalised, we can obtain the angle ϕ as follows:

$$\phi = \cos^{-1}(\vec{N_m} \bullet \vec{N_r}) = \cos^{-1}(X_m X_r + Y_m Y_r + Z_m Z_r).$$

On the other hand, a rotation axis $\vec{n}$ is a cross product of $\vec{N_m}$ and $\vec{N_r}$:

$$\begin{aligned} \vec{n} &= \vec{N_m} \times \vec{N_r} \\ \vec{n}(X_n, Y_n, Z_n) &= \vec{N_m}(X_m,\ Y_m,\ Z_m) \times \vec{N_r}(X_r,\ Y_r,\ Z_r) \\ &= \vec{n}(Y_m Z_r - Z_m Y_r,\ Z_m X_r - X_m Z_r,\ X_m Y_r - Y_m X_r), \end{aligned}$$

where the magnitude of $\vec{n}$ is equal to one.

A `unit quaternion` q corresponding to a rotation angle ϕ and a rotation axis $\vec{n}(X_n,\ Y_n,\ Z_n)$ can be defined by $q(\cos\frac{\phi}{2},\ \vec{n}\sin\frac{\phi}{2})$, where the magnitude of q is identical with one.

Let a `pure quaternion` p corresponding to a vector $\vec{v}$ be $p(0,\ \vec{v})$ which has vector part only. About both a rotation angle ϕ and a rotation axis $\vec{n}$, the rotation of a vector $\vec{v}$ can be achieved by a quaternion multiplication. Namely, the rotation specified by a unit quaternion q transforms a vector $\vec{v}$ into another vector $\vec{v'}$ through the multiplication of three quaternions as follows:

$$\begin{aligned} p'(0, \vec{v'}) &= qpq^{-1} \\ &= (\cos\tfrac{\phi}{2},\ \vec{n}\sin\tfrac{\phi}{2})(0,\ \vec{v})(\cos\tfrac{\phi}{2},\ \vec{n}\sin\tfrac{\phi}{2})^{-1} \\ &= (0,\ \vec{v}\cos\phi + (1-\cos\phi)\vec{n}(\vec{n} \bullet \vec{v}) + \sin\phi(\vec{n} \times \vec{v})), \end{aligned}$$

where $\bullet$ means a dot product of two vectors and $\times$ means a cross product of two vectors.

A rotated vector $\vec{v'}$ is the vector part of this multiplication. The details can be found in *Appendix D*.

3 Rotating a triangle

Our goal is to obtain the original face corresponding to reconstructed normals $\vec{N_r}$'s, starting from an impoverished face specified by model normals $\vec{N_m}$'s. This is achievable by the rotation of each vertex on a triangle, since our impoverished face consists of triangles represented by three vertices.

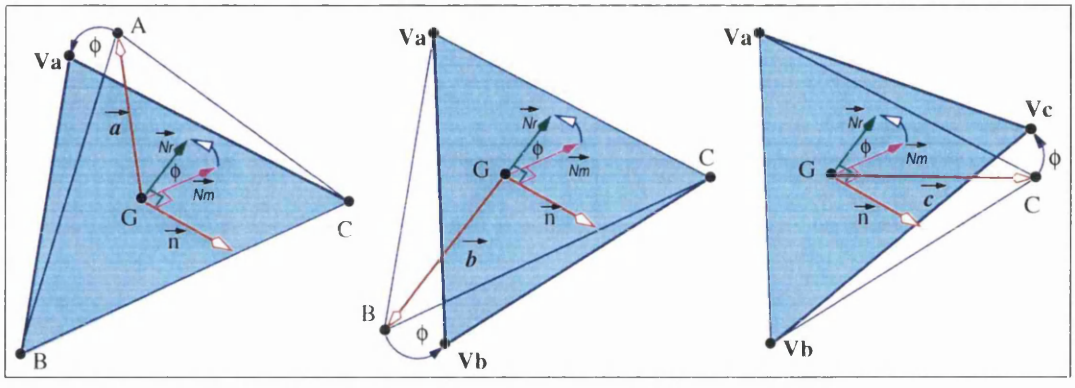


Figure 5.4: Triangle rotation

A unit quaternion q is represented by an angle ϕ and a rotation axis $\vec{n}$ at the gravity centre G of triangle $\triangle ABC$. Each vertex vector can be specified by a pure quaternion p. After applying the quaternion multiplication qpq^{-1} to each vertex vector, a triangle $\triangle ABC$ with a model normal $\vec{N_m}$ is transformed into a triangle $\triangle V_a V_b V_c$ with a reconstructed normal $\vec{N_r}$.

A vertex can be written in the form of a vector from the gravity centre of a triangle. For example, if $A(x_a, y_a, z_a)$ and $G(x_g, y_g, z_g)$ shown in Figure 5.4 are provided, a so-called `vertex vector` $\vec{a}$ is written:

$$\vec{a} = \vec{GA} = (x_a - x_g,\ y_a - y_g,\ z_a - z_g).$$

Generally speaking, a vertex vector $\vec{v}$ makes a pure quaternion $p(0,\ \vec{v})$. In addition, $\vec{N_m}$ and $\vec{N_r}$ make a unit quaternion $q(\cos\frac{\phi}{2},\ \vec{n}\sin\frac{\phi}{2})$. If we apply the quaternion multiplication qpq^{-1} to each vertex of a `model triangle`[1] of an impoverished face, we are able to obtain a `rotated triangle`.

Figure 5.4 illustrates the rotation of a model triangle by the quaternion multiplication, starting from $\triangle ABC$ on the left, with the gravity centre G. Given

[1] A model triangle is oriented by a model normal $\vec{N_m}$ and a rotated triangle is oriented by a reconstructed normal $\vec{N_r}$.

an intensity value I at an image point G, we can obtain a unit quaternion $q(\cos\frac{\phi}{2},\ \vec{n}\sin\frac{\phi}{2})$ from a model normal $\vec{N_m}$ and a reconstructed normal $\vec{N_r}$. For a vertex A, the vertex vector $\vec{a}$ gives a pure quaternion $p(0,\ \vec{a})$. The quaternion multiplication qpq^{-1} rotates the vector $\vec{a}$ so that the vertex A is transformed into a new vertex V_a as shown on the left picture.

If we apply the multiplication to the other vertices B and C, we are able to obtain a rotated triangle $\triangle V_aV_bV_c$, as shown on the right picture. This triangle is now oriented by the reconstructed normal $\vec{N_r}$.

Consequently, our rotating algorithm works perfectly over a model triangle. However, since an impoverished face consists of a number of triangles, the rotation of a model triangle influences the model normals of adjacent triangles. Therefore, whenever a rotation occurs, it is necessary to renew those model normals.

We are going to discuss how we manage the adjacent triangles whenever a triangle is rotated; firstly in 2D and then 3D.

4 Renewing adjacent triangles in 2D

To keep the discussion simple, we are going to do everything in 2D in this section, as shown in Figure 5.5, where the vertical direction indicates the z-axis and the horizontal direction indicates the y-axis. We assume that the original face is the top half of a cylinder and it is approximated by four triangles. The arrows on the original face are surface normals.

A source vector $\vec{L}$, directly toward the northwest is shown in the centre of the figure. The intensity profile shown was created by the *Lambertian* law, which is applied to the interpolated surface normals of the mesh approximation.

Starting from a given set of triangles as an impoverished face, our goal is to find out the approximated mesh of the original face represented by the intensity profile.

From the normal extraction method mentioned in the previous chapter, a reconstructed normal is obtained by three factors: a model normal, an intensity value, and a source vector. A model and reconstructed normal give a unit quaternion to rotate a model triangle. After the rotation, the model normals of the adjacent triangles are renewed, since they have the old information of their model normals. We call this process the `renewing-model-normals`.

Figure 5.6 shows the details of the renewing-model-normals, whenever a tri-

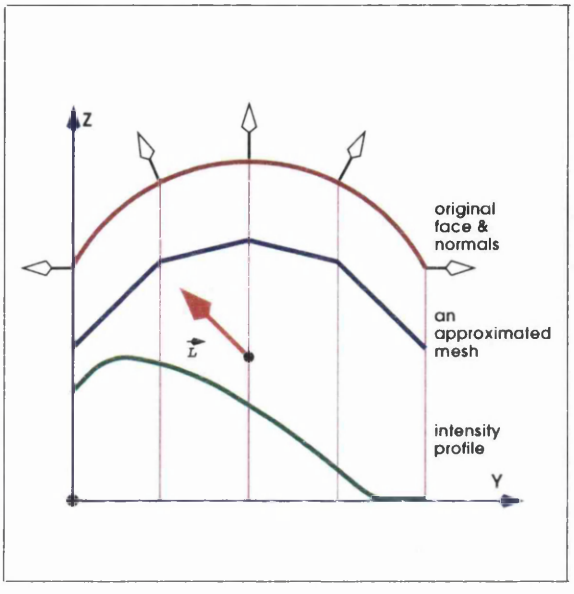


Figure 5.5: An example of surface characteristics in 2D

The original face is the top half of a cylinder and it is approximated by four triangles. There is a source vector $\vec{L}$, directly toward the northwest. The intensity profile is created by the *Lambertian* law applied to the intepolated surface normals of the approximated mesh.

angle is rotated. As shown on the top of the figure, an impoverished face consists of four triangles (T_i, $i = 1$ *to* 4), which is given at the beginning of the process. The family of $\vec{N_m}$ and $\vec{N_r}$ indicate model and reconstructed normals at each stage. Moreover, we assume that an intensity value corresponding to a gravity centre G is available from the intensity profile.

Let us start the renewing-model-normals with the triangle T_1 of the impoverished face. A model normal $\vec{N_{m1}}$ gives a reconstructed normal $\vec{N_{r1}}$. At *stage*1, T_1 is rotated about the gravity centre G_1 so that it is oriented by the reconstructed normal $\vec{N_{r1}}$.

In the figure, a dashed line represents a triangle before being rotated and a shaded circle represents a rotation boundary of a model triangle. The rotation of T_1 influences the geometry of the adjacent triangle T_2. The model normal $\vec{N_{m2}}$ is now renewed to $\vec{N_{m2a}}$. At this point, it is noticable that the model normals of T_3 and T_4 keep their orientations unchanged, and that the rotated triangle T_1 has been *shrunk* along the y-axis.

Using the renewed model normal $\vec{N_{m2a}}$, the reconstructed normal $\vec{N_{r2a}}$ for the triangle T_2 is obtained. At *stage*2, the renewed triangle T_2 is rotated so that it is oriented by $\vec{N_{r2a}}$. This rotation now influences two adjacent triangles T_1 and T_3. Their model normals are renewed by $\vec{N_{m1b}}$ and $\vec{N_{m3b}}$ respectively. Repeatedly, it is noticable that the rotated triangle T_4 is *shrunk* along the y-axis as well.

Similarly, if we apply the renewing-model-normals process to T_3 and T_4 at both *stage*3 and *stage*4, the impoverished face on the top is reconstructed by a face on the bottom, of which triangles are specified by $\vec{N_{m1b}}$, $\vec{N_{m2c}}$, $\vec{N_{m3d}}$, and

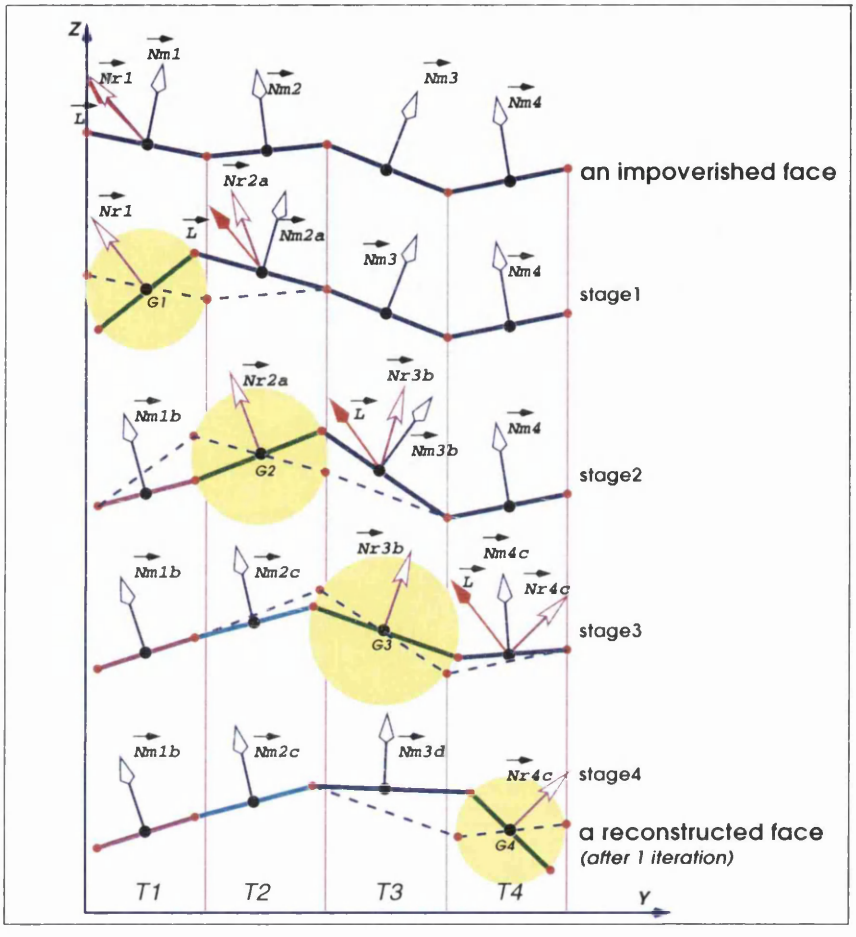


Figure 5.6: Renewing adjacent triangles

$\vec{N_{r4c}}$ respectively.

This reconstructed face, even in only one iteration, has converged into the approximated mesh as shown in Figure 5.5. If we iterate the renewing-model-normals process with the current reconstructed face as the new impoverished face, again and again, it may give the approximated mesh. Moreover if we apply a more dense mesh as an impoverished face, the original face can be obtained with a small error boundary.

Empirically, we found that around ten iterations are usually sufficient to obtain a reconstructed face which is recognisable as the original face. This chapter and the next chapter show a number of experiments illustrating the performance of our face reconstruction method.

4.1 Problems in rotating a triangle

The renewing-model-normals process has problems. Namely when a triangle is rotated, it could be `overlapped` or `intersected` with an adjacent triangle, or it could make an adjacent triangle `overturned`. In addition, the overall size of a reconstructed face could be `shrunk` as shown in Figure 5.6 or be `inflated` as shown in Figure 5.7.

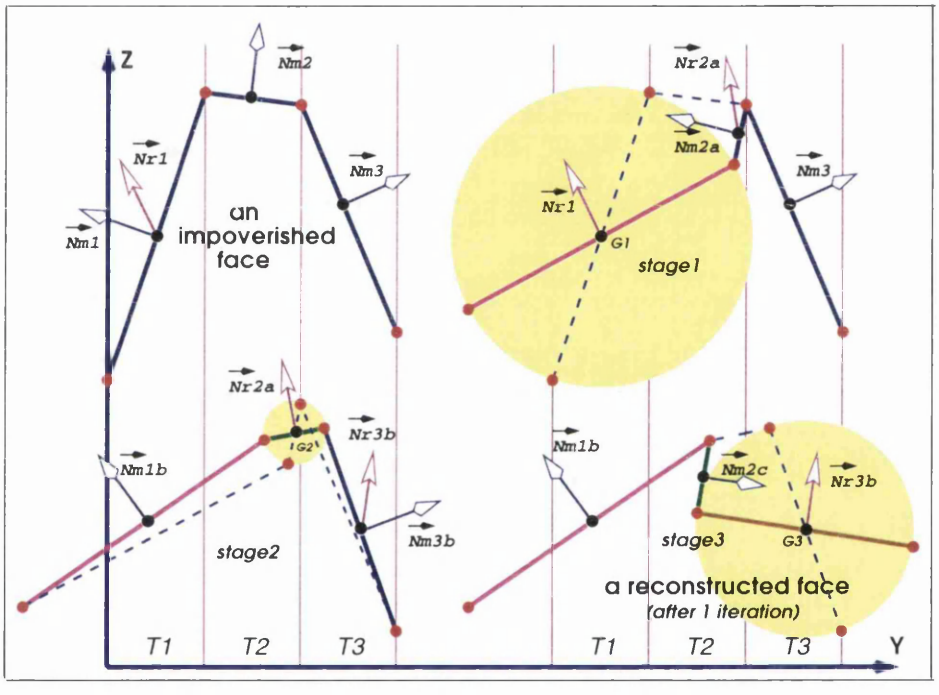


Figure 5.7: Problems in triangle rotation

> After the rotation of T_1, at *stage*1, T_1 is stretched. Similarly, at *stage*3, the rotated triangle T_3 is stretched as well. This act of stretching causes a reconstructed face to be inflated. In addition, the rotation of T_3 makes triangles overlapped one another, and makes T_2 overturned.

Let us understand those problems with an example. Figure 5.7 shows three stages of the renewing-model-normals process mentioned in the previous section. An impoverished face consists of three triangles (T_i, $i = 1$ *to* 3), as shown on the top left. To concentrate on the problems only, we assume that the reconstructed normals are given in advance at each stage.

For the triangle T_1 of the impoverished face, the reconstructed normal $\vec{N_{r1}}$

and the model normal $\vec{N_{m1}}$ rotate T_1 about the gravity centre G_1 so that it is oriented by $\vec{N_{r1}}$ as shown at *stage*1. The rotated triangle T_1 is stretched over the area of the adjacent triangle T_2 and over the other side. It is noticable that the height difference of T_1 along the z-axis was decreased by the rotation.

After the rotation of a boundary triangle, the act of stretching occurs when the height difference is decreased. This stretch causes a reconstructed face to be inflated. The act of shrinking occurs when the height difference is increased, as seen in Figure 5.6.

Because our reconstruction method assumes orthographic projection, the overall size of an impoverished face should be unchanged under the reconstruction. If changed, the correspondence between an image and an impoverished face is not to be trusted iteration by iteration.

In *stage*3, the rotation of T_3 makes triangles overlap one another. T_3 is stretched as well. The overturned triangle T_2 gives totally the wrong information about its model normal, which is now invisible. It causes a reconstructed face at the next iteration to locally move from a correct solution.

On the other hand, after the rotation of a triangle, an intersection may occur when two non-adjacent triangles intersect each other.

4.2 Moving vertices

As mentioned so far, the problems in rotating a triangle make the next iteration unstable and so divergent. There is one possible way to avoid these problems. After the rotation of a model triangle, the y-coordinate values of vertices remain unchanged as before, but only the z-coordinate values are changed. We call this process `moving-vertices`.

Figure 5.8 shows the moving-vertices process, which employs the impoverished face and the reconstructed normals in Figure 5.7. V_x ($x = 1,\ 1a,\ 2,\ 2a$,...) indicates vertices of triangles and G_x ($x = 1,\ 2a,\ 3b$) are the gravity centres of the rotated triangles. A shaded circle represents a rotation boundary of triangles, a thick dashed line along a rotated triangle is removed, and a thin dashed line represents a triangle before being rotated.

After the rotation of T_1 at *stage*1, the vertices are moved so that the y-coordinate values are unchanged and the z-coordinate values are changed. The vertices V_{1a} and V_{2a} show vertices after moving. They are obtained by othographic projection along the z-axis. Namely, the vertices V_1 and V_2 are projected onto the line passing through the rotated vertices.

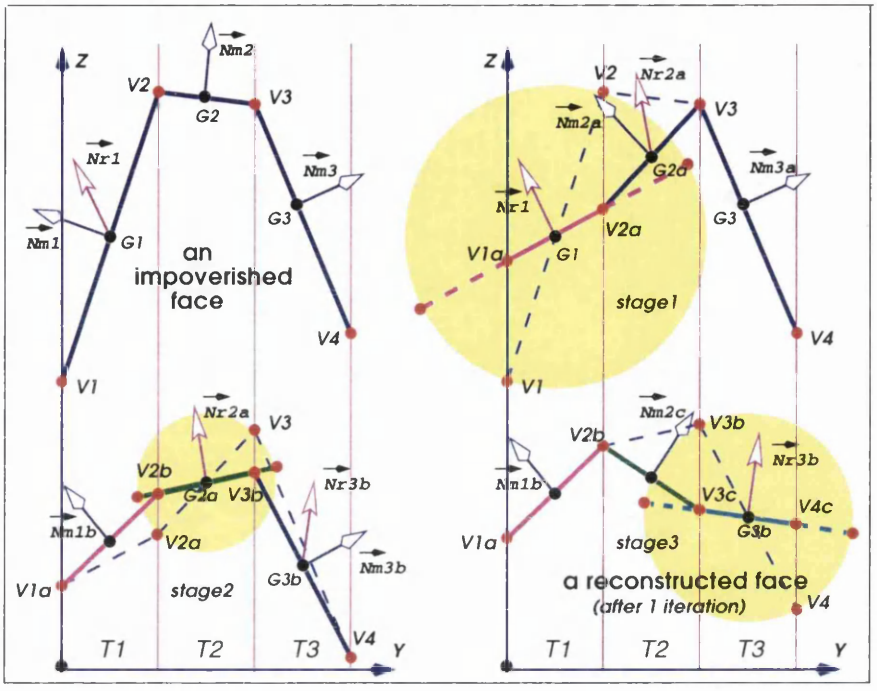


Figure 5.8: Moving vertices

The thin dashed lines show the triangles before rotation, the thick dashed lines show the clipped triangles whose vertices are moved into the boundary of each triangle, and the thick solid lines represent triangles applied by the moving-vertices method at every stage.

The triangle T_1 consisting of the moved vertices is not inflated, but it is still orientated by the reconstructed normal $\vec{N_{r1}}$. Secondarily, T_2 renews its model normal with $\vec{N_{m2a}}$, which consisting of two vertices V_{2a} and V_3. $\vec{N_{r2a}}$ is a reconstructed normal for T_2.

After the rotation of T_2 at *stage*2, the moved vertices V_{2b} and V_{3b} are obtained by the same projection of V_{2a} and V_3. T_2 is oriented by $\vec{N_{r2a}}$. T_1 and T_3 are renewed by $\vec{N_{m1b}}$ and $\vec{N_{m3b}}$ respectively. $\vec{N_{r3b}}$ of T_3 consisting of V_{3b} and V_4 is obtained from $\vec{N_{m3b}}$.

At *stage*3, in the similar way, the triangle T_3 has the moved vertices V_{3c} and V_{4c} and it is oriented by $\vec{N_{r3b}}$. T_2 is renewed by a model normal $\vec{N_{m2c}}$. The solid lines in this stage show a reconstructed face after one iteration.

The moving-vertices process allows a reconstructed face to avoid the problems occuring when triangles are rotated. In this thesis, the moving-vertices together

with the renewing-model-normals process are employed whenever we reconstruct an original face. The next section will discuss the moving-vertices process in 3D space more in detail.

5 Moving vertices in 3D

5.1 A classification of vertices used

In reconstructing an original face, a triangle rotation of an impoverished face suffers from problems such as overlap, intersection, overturn, shrinkage, and inflation. To avoid them, we have developed the moving-vertices process as mentioned in the previous section. In this section, we will explain how the vertices of a rotated triangle are moved in 3D space.

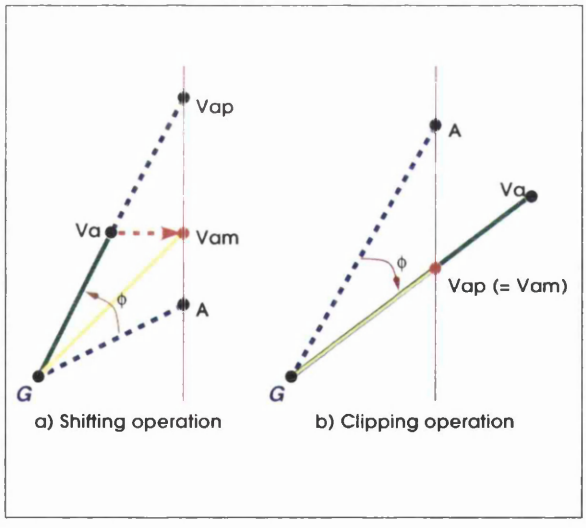


Figure 5.9: Determining a moved vertex

G is a gravity centre of a triangle. A shows one of three vertices before rotated, called a `model vertex`. V_a is a `rotated vertex`, and V_{ap} is a `projected vertex` created by projecting A onto a plane which contains a rotated triangle. V_{am} is a `moved vertex` which determines a reconstructed triangle.

In Figure 5.9, G is a gravity centre of a triangle, and A is one of three vertices before rotated. We will call a vertex before being rotated a `model vertex`. After a model vertex is rotated, it becomes a `rotated vertex` such as V_a. V_{ap} is a `projected vertex`, which is created by projecting A onto a plane containing the rotated triangle. V_{am} is a `moved vertex` which determines a reconstructed triangle after rotation.

5.2 Defining operations

There are two operations to determine a moved vertex: one is a `shifting` operation and the other is a `clipping` operation. The shifting operation is applied when, from the gravity centre G, the distance of a model vertex A ($\overline{GA} = \overline{GV_a}$) is shorter than that of a projected vertex V_{ap} ($\overline{GV_{ap}}$). In this case, the z-coordinate (height) of a rotated vertex V_a becomes that of a moved vertex V_{am}.

On the other hand, the clipping operation is applied when, from G, the distance of A ($\overline{GA} = \overline{GV_a}$) is longer than that of V_{ap} ($\overline{GV_{ap}}$). The z-coordinate (height) of V_{ap} becomes that of V_{am}.

5.3 Determining a reconstructed triangle

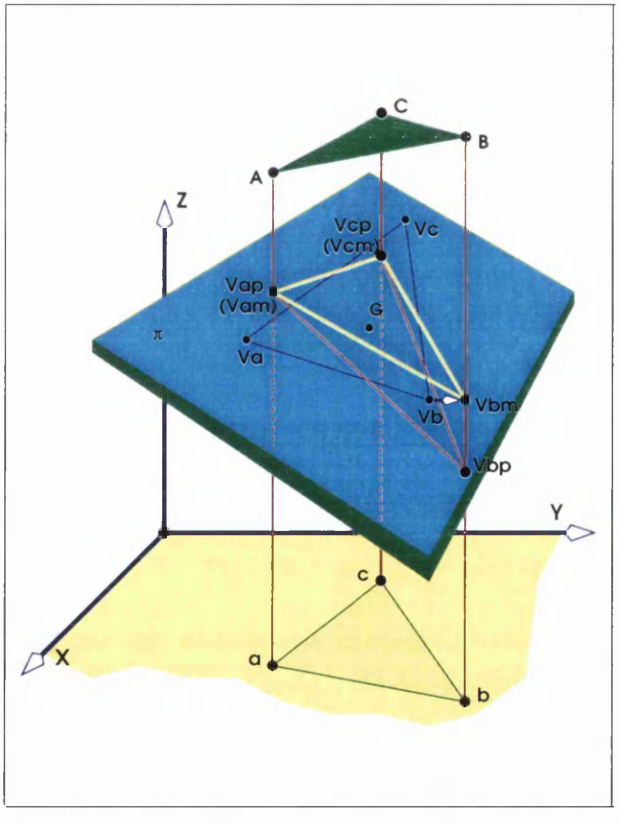


Figure 5.10: Determining a reconstructed triangle

G is a gravity centre of a rotated triangle $\triangle V_aV_bV_c$ and also a model triangle $\triangle ABC$. The model triangle is elevated only to explain the two operations better. $\triangle V_{ap}V_{bp}V_{cp}$ is a projected triangle which is a projection of $\triangle ABC$ onto a plane π which contains the rotated triangle. $\triangle abc$ is a xy-projected triangle of $\triangle ABC$ (or $\triangle V_{ap}V_{bp}V_{cp}$ or $\triangle V_{am}V_{bm}V_{cm}$). Finally, $\triangle V_{am}V_{bm}V_{cm}$ is a reconstructed triangle.

Let us apply those two operations to a model triangle to determine a reconstructed triangle in 3D space. Figure 5.10 shows a plane π and five triangles referenced in determining a reconstructed triangle as follows:

- a model (impoverished) triangle ($\triangle ABC$),

- a rotated triangle ($\triangle V_aV_bV_c$),
- a projected triangle ($\triangle V_{ap}V_{bp}V_{cp}$),
- a reconstructed triangle ($\triangle V_{am}V_{bm}V_{cm}$), and
- a xy-triangle ($\triangle abc$) which is a projection of the model triangle $\triangle ABC$ onto xy-plane.

The projections of three triangles ($\triangle ABC$, $\triangle V_{ap}V_{bp}V_{cp}$, and $\triangle V_{am}V_{bm}V_{cm}$) onto the xy-plane are identical. It should be said that $\triangle ABC$ has the same gravity centre G of $\triangle V_aV_bV_c$, but it is elevated to explain the two operations better in Figure 5.10.

Starting from a model triangle $\triangle ABC$, we can obtain a rotated triangle $\triangle V_aV_bV_c$ by applying the quaternion multiplication to each vertex vector of the model triangle. Let the rotated vertices be $V_a(X_1,\ Y_1,\ Z_1)$, $V_b(X_2,\ Y_2,\ Z_2)$, and $V_c(X_3,\ Y_3,\ Z_3)$. The plane π containing $\triangle V_aV_bV_c$ can be described in the form as follows:

$$\pi :\ Dx + Ey + Fz = H$$

where (x, y, z) is any point on the plane π, and the coefficients D, E, F, and H are constants describing the spatial properties of the plane.

We can obtain those coefficients by solving a set of three plane equations using coordinate values for three rotated vertices V_a, V_b, and V_c and also an equation using the *Cartesian* components (D, E, F) as the surface normal vector of the plane. The equations for obtaining the coefficients can be written as:

$$\begin{aligned}
D &= y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \\
E &= z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2) \\
F &= x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \\
H &= x_1(y_2z_3 - y_3z_2) + x_2(y_3z_1 - y_1z_3) + x_3(y_1z_2 - y_2z_1).
\end{aligned}$$

Now we can obtain the projected vertices V_{ap}, V_{bp}, and V_{cp} in terms of the model vertices $A(X_a, Y_a, Z_a)$, $B(X_b, Y_b, Z_b)$, and $A(X_c, Y_c, Z_c)$ and the coefficients as follows:

$$\begin{aligned}
V_{ap}(X_{ap},\ Y_{ap},\ Z_{ap}) &= (X_a,\ Y_a,\ \frac{H - DX_a - EY_a}{F}), \\
V_{bp}(X_{bp},\ Y_{bp},\ Z_{bp}) &= (X_b,\ Y_b,\ \frac{H - DX_b - EY_b}{F}), \\
V_{cp}(X_{cp},\ Y_{cp},\ Z_{cp}) &= (X_c,\ Y_c,\ \frac{H - DX_c - EY_c}{F}).
\end{aligned}$$

In Figure 5.10, $\overline{GV_a}$ ($\overline{GV_c}$) is longer than $\overline{GV_ap}$ ($\overline{GV_cp}$) and $\overline{GV_b}$ is shorter than $\overline{GV_bp}$. By applying two operations, the moved vertices V_{am}, V_{bm}, and V_{cm} for the reconstructed triangle $\triangle V_{am}V_{bm}V_{cm}$ are determined as:

$$
\begin{aligned}
V_{am}(X_{am},\ Y_{am},\ Z_{am}) &= (X_a,\ Y_a,\ \frac{H - DX_a - EY_a}{F}),\\
V_{bm}(X_{bm},\ Y_{bm},\ Z_{bm}) &= (X_b,\ Y_b,\ Z_2),\\
V_{cm}(X_{cm},\ Y_{cm},\ Z_{cm}) &= (X_c,\ Y_c,\ \frac{H - DX_c - EY_c}{F}).
\end{aligned}
$$

5.4 Performance of the moving-vertices process

We have explored how to determine a reconstructed triangle, using the moving-vertices process. If we apply this process to all triangles of an impoverished face, then we obtain a reconstructed face. It does not suffer from the problems occurring due to the triangle rotation.

Figure 5.11 demonstrates the performance of this moving-vertices process. The upper left illustrates a reconstructed face without employing the moving-vertices process. It shows the problems caused by the triangle rotation, as mentioned in the previous section. These appear as turbulent black-spots on the face and cause protrusions on the boundaries. One of these problem areas is shown in the left bottom of the figure. This is a magnified mesh of the rectangular part of the reconstructed face. It shows clearly that some rotated triangles are overlapped and intersect one another. others are overturned, shrunk, and inflated. These problems are due to model triangles. They are too steep to be visible in the impoverished face, however the corresponding intensity values of the input face image are not zero (black).

On the other hand, the upper right in Figure 5.11 shows a reconstructed face generated by applying the moving-vertices process. The turbulent black-spots and protrusions are removed. The right bottom shows a mesh magnifying the rectangular area of the reconstructed face.

In our face reconstruction method, this moving-vertices process in 3D is applied whenever the rotation of a triangle occurs. And then the renewing-model-normals process is followed. From this point, all reconstructed faces are obtained by applying the moving-vertices process.

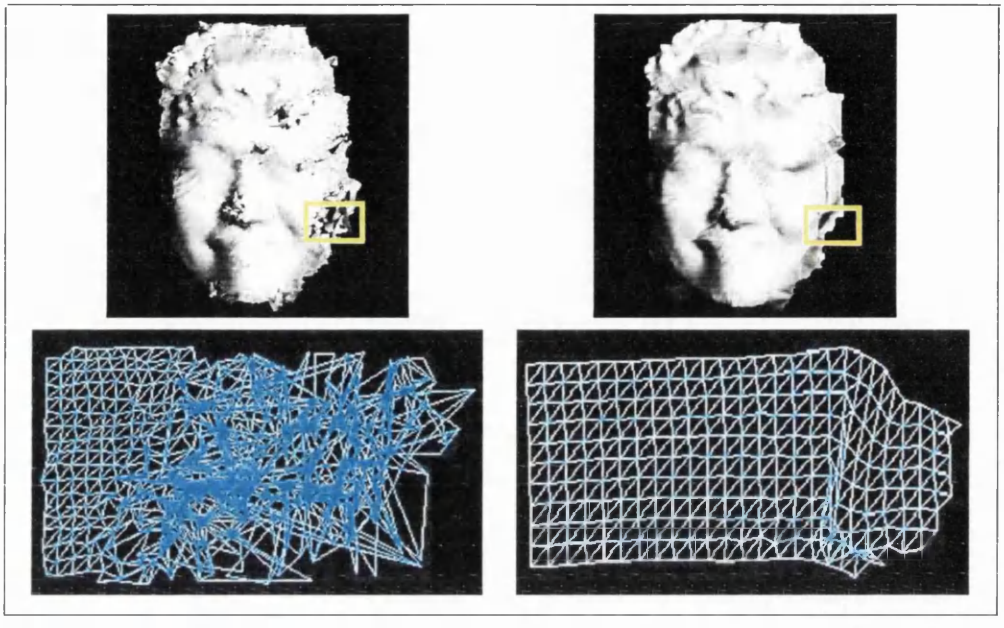


Figure 5.11: Reconstruction with/without the moving-vertices process

The upper left illustrates a reconstructed face without the moving-vertices process, while the upper right shows it with that process. The lower pictures represent the rectangular areas on the upper faces. They are magnified in the form of mesh.

6 A Reconstruction Example : from a very impoverished face

At this point, we have fully discussed the main flow of operations in our face reconstruction process shown in Figure 5.2. In this section, in order to demonstrate one result, we will use the impoverished face and face image, shown in Figure 5.1.

6.1 Shading : input face image and source vector

The input face image shown in Figure 5.1 is a synthetic rendering of a face image by applying the *Lambertian* law with a source vector $\vec{L}(0.40, 0.21, 0.89)$. In fact most *SFS* techniques use a source vector illuminated in the front such

as $\vec{L}(0,0,1)$, so that input images do not include any self-shadows. It should be stressed that here the source vector $\vec{L}$ is quite different from that usually employed in *SFS* research. Namely, the original images include self-shadows.

An individual intensity I of an image together with a source vector $\vec{L}$ determines a radius R of an intensity cone in space, as mentioned in the previous chapter. This helps us to extract a reconstructed normal for a triangle.

6.2 Prior knowledge : impoverished face

In *Chapter 3*, we discussed the method for obtaining impoverished faces. The impoverished face in Figure 5.1 preserves the global topology of the original face in the input face image, but the surface details such as the nose, mouth, and eyes are very distorted. What is worse, it may no longer be possible to be recognised as the original. The lost details are however, reconstructed from the input face image.

The impoverished face was first deprived of 90% of its original triangles. The remaining triangles were then slightly and randomly rotated to be more distorted. Finally, it was then remeshed by sampling height fields at uniform intervals of two pixels. The height fields were obtained using linear interpolation. The impoverished face in Figure 5.1 is a shaded image of the remeshed face. It represents less than 10% of the structure of the original face.

6.3 Face reconstruction

Face reconstuction is the iterative act of restoring impoverished faces to the original faces, using the information obtained from face images in the form of shading.

Figure 5.12 shows a sequence of reconstructed faces generated by applying our method as iteration goes on. The top left-hand side is the impoverished face. The others are reconstructed faces in 10 iterations[2] from left to right and downwards. For example, the top-middle face was obtained at the first iteration and the bottom-middle one at the 10^{th} iteration. The reconstructed face in Figure 5.1 was also obtained at the 40^{th} iteration. We found that around ten iterations are usually sufficient to obtain a reconstructed face that could be recognised.

[2]In the thesis, the number of iterations does not always mean that a reconstructed face implies a minimum error. The discussion about iteration will be given in the next chapter.

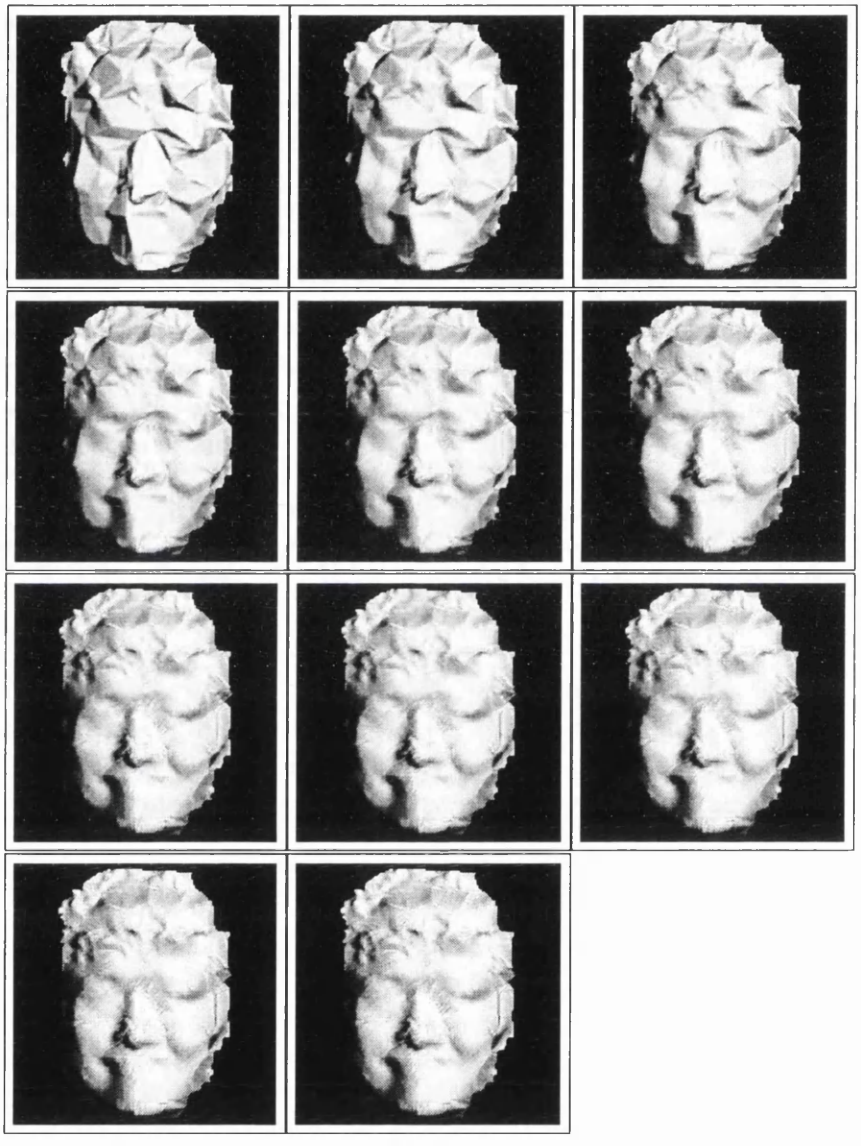


Figure 5.12: A sequence of reconstructed faces with the *moving-vertices* process

The top left is the impoverished face in Figure 5.1. The others are a sequence of reconstructed faces in 10 iterations from left to right and downwards. For example, the top-middle face was obtained by one iteration and the bottom-middle one was generated by ten iterations.

As seen in the reconstruction sequence, our method gives a robust means for reconstructing the original face of a single image, starting from an impoverished face. However, some bright lines (*furrows*) are created as the iteration goes on. These furrows look like *Mach* bands [Rat72, Pho75]. We believe that they are due to moving vertices. Empirically, they are created in the areas of an impoverished face that give bad model-normals in the reconstruction process.

7 Discussion

SFS techniques recover 3D shape of the original object that has been transformed into a 2D image. There have been many techniques proposed up to now, however the recovered shape is only a needle diagram [Hor75, WH97], which is a collection of reconstructed normals.

Using an assumption of smoothness, they reconstruct the original surface from the needle diagram. Unfortunately, the reconstructed surfaces suffer from the flatness problem. That is, they cannot be seen in other directions as presented in *Chapter 2.*

In this chapter, we have presented a robust method for reconstructing the original face in a single image using a very impoverished face in the form of a set of triangles. As discussed in the previous chapter, an intensity value at a particular image point gives an intensity cone, assuming that there is a source vector.

From a standpoint of *SFS*, most methods have identified a surface normal out of the intensity cone using additional constraints. However we employ a model normal of the impoverished face corresponding to the image point in order to decide a surface normal. This model normal is regarded as prior knowledge in the thesis.

At this point, if a model triangle specified by a model normal can be transformed into a triangle specified by a reconstructed normal, the reconstruction is locally achievable. To do so, we introduced a quaternion multiplication. However, the rotation of a model triangle may result in geometric problems maintaining links with adjacent triangles.

To avoid these problems, we apply the `moving-vertices` process to the rotated triangle, which is endowed with two operations; a shift and a clipping operation. The moving-vertices process gives a reconstructed triangle, which is not an exact solution but an approximation. The model normals of triangles adjacent to the reconstructed triangle are, at this point, renewed.

A whole set of reconstructed triangles obtained in this way is a reconstructed face. If we apply our method to the reconstructed face iteratively, we can obtain as acceptable a face as the original, even if a very impoverished face is given to start with. Especially, as a remarkable result, we can see the reconstructed face in the different direction as shown in Figure 5.1.

Chapter 6

Analysis of Reconstructed Faces

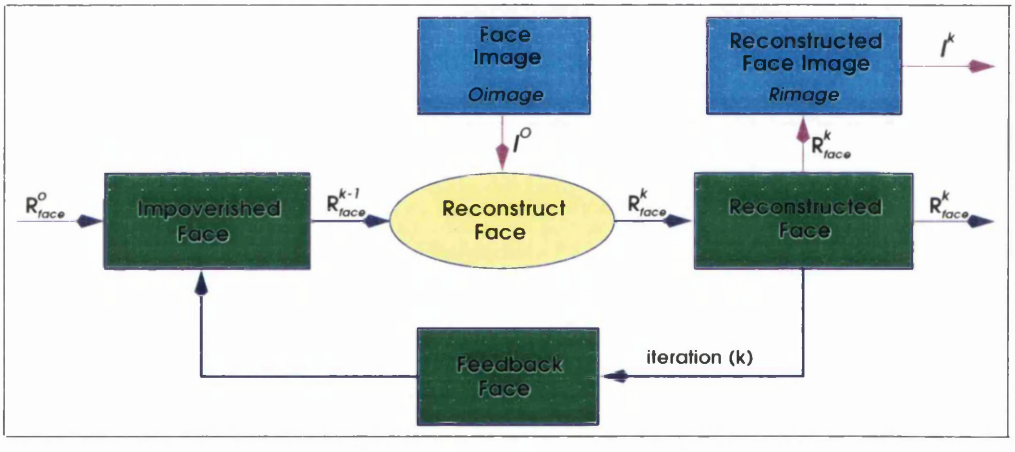


Figure 6.1: Iteration cycle under the face reconstruction

In the previous chapter, we have discussed the face reconstruction method in detail, with an example of a very impoverished face shown in Figure 5.1. As seen in Figure 5.12, in order to produce a well-shaped face, it is necessary to iterate the reconstruction process.

Figure 6.1 shows a cycle of iterations under the face reconstruction. R^0_{face} represents an impoverished face for the first iteration and R^{k-1}_{face} for the $(k-1)^{th}$ iteration. In addition, R^k_{face} is a reconstructed face, which is returned as a new impoverished face for the next iteration.

Now we can describe the original face O_{face} in an image as a reconstructed face R^k_{face} with an error face E^k_{face} with respect to every iteration;

$$O_{face} = R^k_{face} + E^k_{face}.$$

In this chapter, we are going to analyse the error term. Two statistical estimators are defined here according to each iteration: average and standard deviation of intensity differences between two images. These two images are called `Oimage` and `Rimage`. The former is for an input face image and the latter is for the image obtained from a reconstructed face.

Using the statistical estimators, the reconstructions from differently impoverished faces are analysed. A reconstruction album is then presented. This includes the reconsturctions from various faces.

1 Error Estimators for Reconstructed Faces

Statistically, measures of location and variability may serve to characterise a set of data and convey some of its salient features [Dev87]. For our purpose, two statistical estimators are defined in terms of arithmetic average and standard deviation for a given set of data.

Before defining these estimators, it is convenient to denote data obtained under the reconstruction process as shown in Figure 6.1:

- I_{ij}^{O} : an intensity value at a particular point (i, j) in the *Oimage*.
- R_{face}^{k} : a reconstructed face in the form of mesh, which is returned as a new impoverished face for the next iteration.
- I_{ij}^{k} : an intensity value at a particular point (i, j) in the *Rimage*, which is obtained from R_{face}^{k}.
- e_{ij}^{k} : an intensity difference between I_{ij}^{O} and I_{ij}^{k}, which is an error term at a particular image point.

In the above notations, i and j vary according to the width and height of images and k represents the k^{th} iteration.

1.1 Average of intensity differences

Given a set of intensity differences $\{e_{00}^k,\ e_{01}^k,\ e_{02}^k, ...\}$ at the k^{th} iteration, we define an average error term as follows:

$$E_{image}^k = \frac{1}{N}\sum_{i=1}^{w}\sum_{j=1}^{h} e_{ij}^k$$

$$= \frac{1}{N}\sum_{i=1}^{w}\sum_{j=1}^{h} |I_{ij}^O - I_{ij}^k|,$$

where N is the number of pixels in an image[1], $\sum$ is summation, and $|...|$ is an absolute value.

The estimator E_{image}^k so-called *AVID*[2] is an arithmetic average of intensity differences D_{ij}^k between two images: one is the *Oimage* consisting of I_{ij}^O's, and the other is the present *Rimage* consisting of I_{ij}^k's.

The values of the *AVID* vary from 0 to 255, because the scale of intensity is 256 in this thesis. A value of 0 means that the present *Rimage* is identical with the *Oimage*. Namely, they have no intensity differences. Whereas higher values indicate that the present *Rimage* has more errors in intensity compared with the *Oimage*.

The *AVID* can be referred to the average error term for a reconstructed face at the k^{th} iteration. If a reconstructed face has a value of 0, then it is identical with the original face without error. Using this estimator, therefore we can decide convergence criteria for the reconstruction process. If the values of *AVID*s become smaller as the iteration goes on, reconstructed faces converge into the original face. If a value of the *AVID* is minimum, the reconstructed face has a minimum error.

1.2 Standard deviation of intensity differences

If two images have the same *AVID* values, then dispersion metrics must be used to distinguish between them. We employ a standard deviation of a set to represent the degree of dispersion.

According to the same set used in defining the *AVID*, we define a standard

[1] If the width and height of an image are identical with w and h respectively, then N is equal to $w \cdot h$.

[2] The *AVID* stands for AVerage Intensity Differences between *Oimage* and *Rimage*.

deviation as follows:

$$S^k_{image} = \sqrt{\frac{1}{N}\sum_{i=1}^{w}\sum_{j=1}^{h}(D^k_{ij} - E^k_{image})^2}$$

$$= \sqrt{\frac{1}{N}\sum_{i=1}^{w}\sum_{j=1}^{h}(D^k_{ij})^2 - (E^k_{image})^2}.$$

The estimator S^k_{image} so-called *SDID*[3], is a measure of dispersion of intensity differences about E^k_{image} (*AVID*). A small value of *SDID* means that most D^k_{ij}'s are close to the *AVID*. That is, there is a relatively small amount of dispersion in the set. On the other hand, a large value implies that most D^k_{ij}'s are far from the *AVID*. Therefore, when a value of one *SDID* is smaller than that of the other, a reconstructed face may be optimum.

2 Reconstruction from Differently Impoverished Faces

As mentioned in *Chapter 3*, prior knowledge can be represented by an impoverished face. This refers to a degree of information about the original face. For example, an 80% impoverished face represents 20% information.

How much information about the original face is sufficient to obtain a reconstructed face? To answer it, we applied the reconstruction method to differently impoverished faces.

Figure 6.2 shows the results of reconstructed faces from differently impoverished faces. The left-hand side shows an input face image with a source vector $\vec{L}(0.40, 0.21, 0.89)$.

The middle columns represent differently impoverished faces. They are 20%, 40%, 60%, and 80% impoverished from top to bottom. We call them *me20*, *me40*, *me60*, and *me80* respectively. Facial features gradually disappear as the impoverishment goes on. The *me80* is hardly recognisable as the original face in the input image. It preserves the global topology of the original face but the surface details, such as the nose, mouth, and eyes, are lost.

The right columns are reconstructed faces, which are obtained by 7, 8, 9, and 16 iterations of the reconstruction process. Figure 6.3 shows a sequence of

[3]The *SDID* stands for Standard Deviation of Intensity Differences between *Oimage* and *Rimage*.

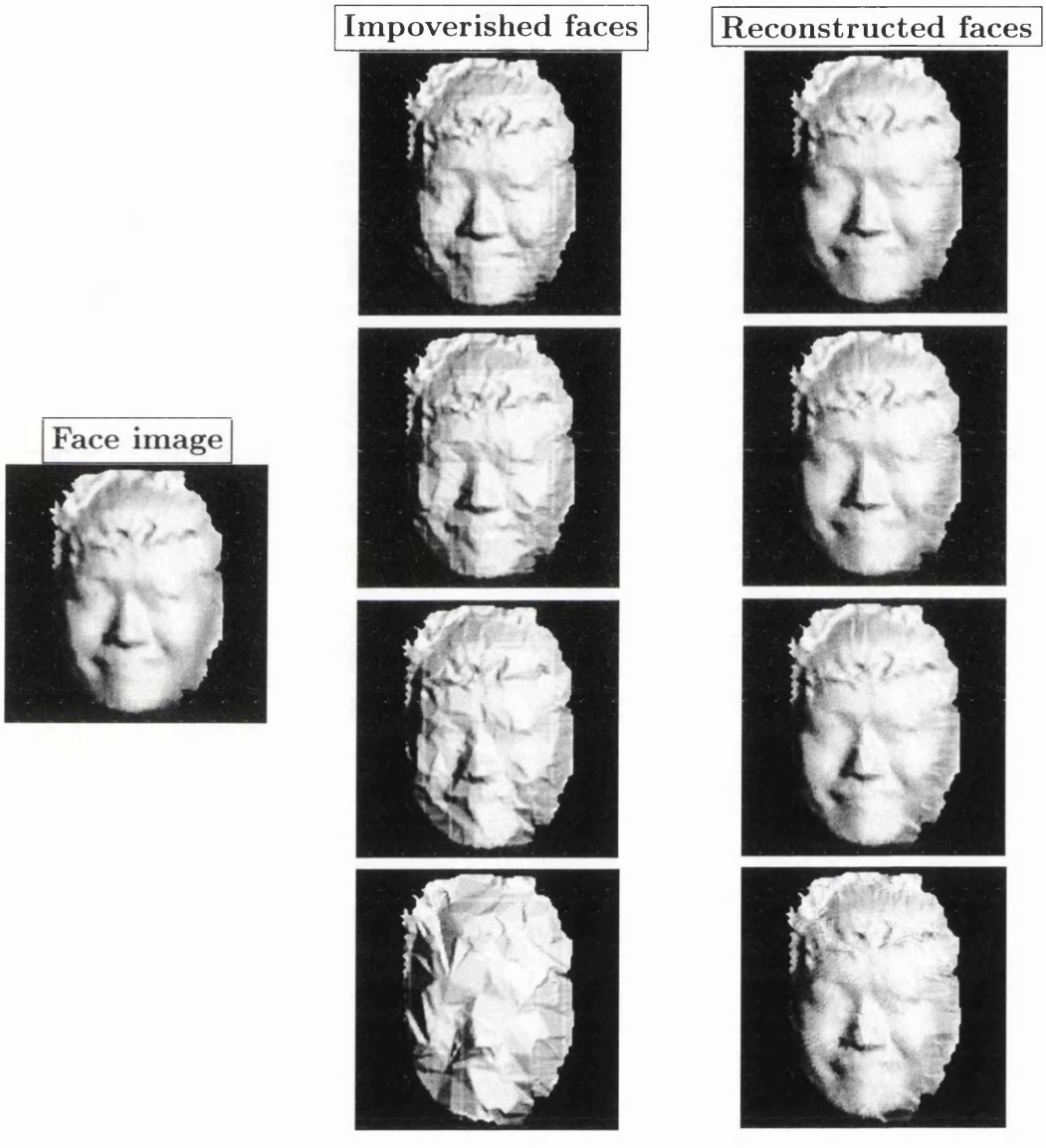


Figure 6.2: Reconstruction from differently impoverished faces

The left-hand side shows an input face image with a source vector $\vec{L}(0.40, 0.21, 0.89)$. The middle columns represent differently impoverished faces. They are 20%, 40%, 60%, and 80% impoverished from top to bottom. The right columns are reconstructed faces which are obtained by 7, 8, 9, and 16 iterations of the reconstruction process.

Figure 6.3: A sequence of reconstructed faces from the *me80*

The top left is the 80% impoverished face in Figure 6.2. The first three rows represent the reconstructed faces obtained in 11 iterations from left to right and downwards. The bottom rows show the reconstructed face at the 16^{th} iteration. It is viewed in the different directions.

reconstructed faces from the *me80*. The top left is the 80% impoverished face in Figure 6.2. The first three rows represent the reconstructed faces obtained in 11 iterations from left to right and downwards. The bottom rows show the reconstructed face at the 16^{th} iteration. This is viewed in different directions.

The iteration number for an optimal reconstructed face is decided by the statistical estimators defined in the previous section. Together with the 90^{+}% impoverished face in *Chapter 5*, an analysis for the *me20*, *me40*, *me60*, and *me80* is focused on the *AVIDs* and *SDIDs* obtained by 40 iterations. We call the 90^{+}% impoverished face *me90Morp*.

2.1 Analysis with averages of intensity differences

Figure 6.4 shows the curves plotted by the *AVID* values corresponding to each iteration. Exact values can be found on Table E.1 in *Appendix E*. The slope of each curve depends on how much a face is impoverished. The curves for the 20% to 80% impoverished faces have minimum values that gradually increase with the number of iterations. They eventually reach a minimum *AVID* feature. From this we can reconstruct an optimal face with a minimum error, as shown in Figure 6.2. A face with around 20% prior knowledge of the original face can be reconstructed so that it has a minimum error.

On the other hand, the graph corresponding to the *me90Morp* decreases in 40 iterations, as shown on the bottom in Figure 6.4. We cannot decide an optimal reconstructed face from the 90^{+}% impoverished face. In this case, we may decide a reconstructed face that is recognisable. For example, amongst the reconstructed faces shown in Figure 5.12 in *Chapter 5*, the face at the 10^{th} iteration can be chosen.

Of course, we might find an optimal face from the *me90Morp*, if we apply the iteration again and again. However, as mentioned in *Chapter 5*, the reconstructed faces have furrows[4]. These appear as bright lines on the reconstructed images. So, in this thesis, we choose the reconstructed faces around the 10^{th} iteration. Not because they have minimum errors, but because the results can be recognised as the original faces.

Table 6.1 summarises the characteristics of the *AVIDs* concerning variously impoverished faces from the *me20* to *me90Morp*. The numbers in parentheses represent the number of iterations when the reconstructed faces are produced. For example, the *me20* starts with an error term 7.49 as a maximum and creates a reconstructed face with a minimum error 5.05 at the 7^{th} iteration. The *me40*

[4]We will address one possible way to avoid this problem in the final chapter.

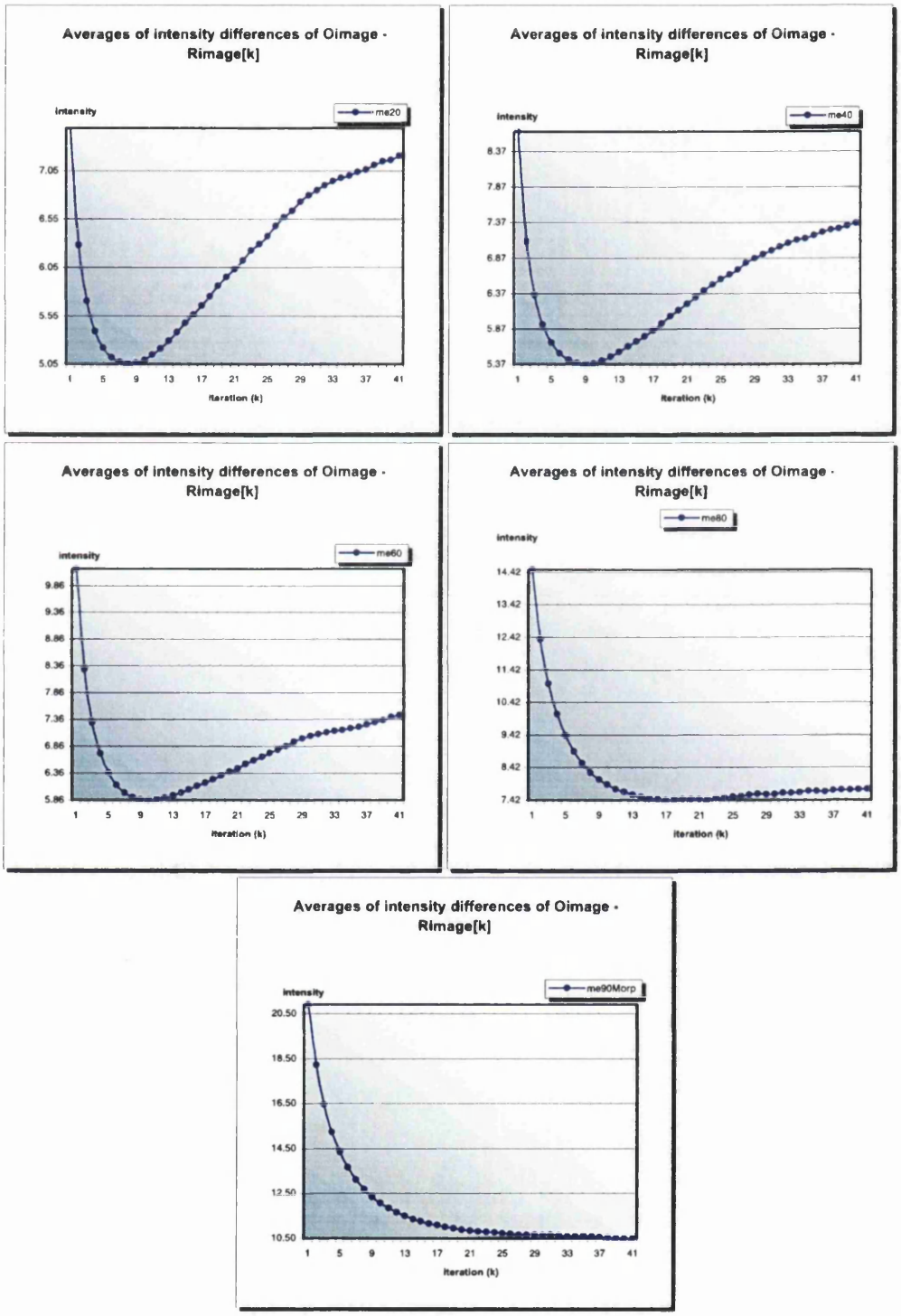


Figure 6.4: Graphs of *AVID*s about variously impoverished faces

	me20	me40	me60	me80	me90Morp
max	7.49	8.65	10.17	14.50	20.21
min	5.05 (7)	5.37 (8)	5.86 (8, 9)	7.42 (16)	N/A
const	N/A	N/A	N/A	N/A	10.50 (40)

Table 6.1: Summaries of of *AVIDs* about variously impoverished faces

starts from a maximum error 8.65 and creates a minimum error 5.37 at the 8^{th} iteration. Similarly, the *me90Morp* has a maximum 20.21 and a decreasing constant 10.50 at the 40^{th} iteration.

In general, we believe that the number of iterations to reach a minimum error increases according to the given impoverished face.

2.2 Analysis with standard deviations of intensity differences

In practice, we found that curves of *SDIDs* in Figure 6.5 are very similar to those of *AVIDs*, as shown in Figure 6.4. The details can be found on Table E.1 in *Appendix E*. Again, those curves dropped rapidly at the beginning of each iteration. The shapes of curves for the 20% to 80% impoverished faces touch minimum values and then gradually increase according to the number of iterations. The increasing slopes become horizontal as the impoverished levels go higher. This says that the error dispersions of reconstructed faces become smaller, the so-called `stable` region, and then become larger, the so-called `unstable` region.

The reconstructed face of the *me60* was obtained at the 9^{th} iteration when its *SDID* is minimum. At this point, it should be said that the minimum values of the *SDIDs* in Figure 6.5, except the bottom curve, were created when the corresponding *AVIDs* are nearly minimum.

On the other hand, the curve for the *me90Morp* gradually approaches a constant 25.78, as shown on the bottom in Figure 6.5. Although we cannot decide the minimum value, it is clear that the reconstructed faces are stable after 40 iterations. In addition, it is expected that the reconstructed faces starting from the *me90Morp* have at least an error dispersion around the constant value.

Table 6.2 summarises the characteristics of the *SDIDs* for the variously impoverished faces. The numbers in parentheses represent the number of iterations when the resultant faces were produced. For example, the *me20* starts with 19.34

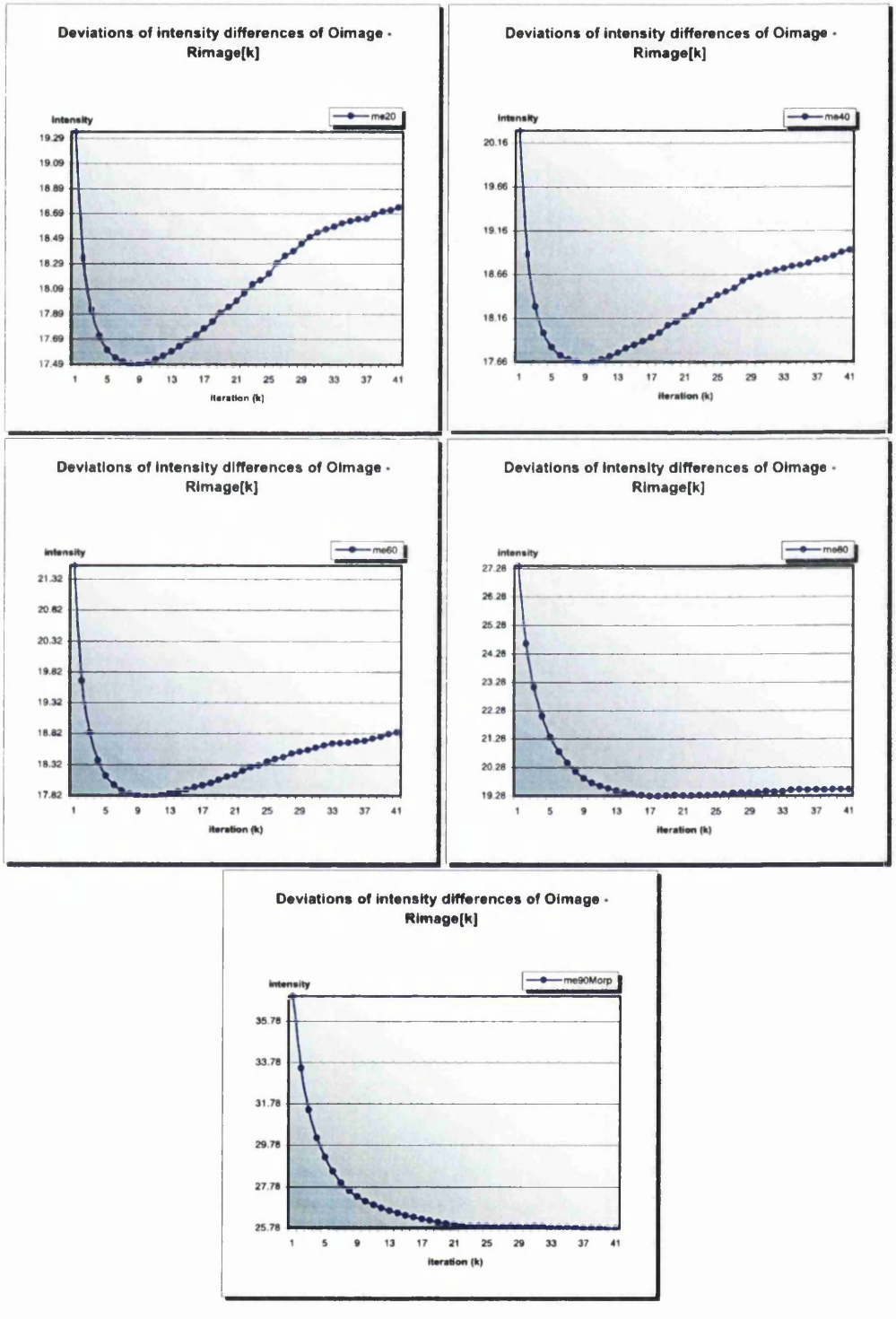


Figure 6.5: Graphs of *SDID*s according to variously impoverished faces

	me20	me40	me60	me80	me90Morp
max	19.34	20.30	21.54	27.37	36.99
min	17.49 (8)	17.66 (7, 8)	17.82 (9, 10)	19.28 (16, 17)	N/A
const	N/A	N/A	N/A	N/A	25.78 (39, 40)

Table 6.2: Summaries of of *SDID*s about variously impoverished faces

as a maximum *SDID* and produces a reconstructed face with a minimum of 17.49 at the 8^{th} iteration. The *me40* implies a maximum *SDID* 20.30 at the beginning and results in a minimum of 17.66 at the 7^{th} and 8^{th} iteration. Similarly, the *me90Morp* has a maximum *SDID* 36.99 and a constant of 25.78 at the 39^{th} and 40^{th} iteration.

In general, we believe that the number of iterations in order to reach a minimum dispersion of errors increases according to the impoverished face. This is similar to the characteristics for the *AVID*s.

3 Reconstruction Album

Until now we have described the performance of our reconstruction method for differently impoverished faces of a person. In this section, we apply our method to four different objects. These include two human faces, a hand, and an instep, as shown in Figure 6.6. The original objects are also adapted from *C3D* of *The Turing Institute.*

On the other hand, as a worst example of the reconstruction, if we have no information about the original face, then we cannot provide impoverished faces as prior knowledge. How can we reconstruct an approximation of an original face without an impoverished face? It is still possible to do this. Our reconstruction method uses a generalised simple face instead of an impoverished face[5].

This question also demonstrates the possibility of using a half-cylinder[6] and a flat surface as the simple face in *Album2* and *Album3.* If it is possible to obtain an approximation of the original face starting from a half-cylinder or flat

[5] An impoverished face is an abstracted version based on the original face, while a generalised simple face has very little connection with the original one such as a half-cylinder, half-ovoid, or flat surface.

[6] We can imagine a half-ovoid as another kind of simple faces. In practice, their results are similar to each other. In *Album2*, they will be discussed.

surface, our method may lead to a way of reconstructing the original face from arbitrary face images.

3.1 Album1 : from four different objects

Figure 6.6 shows the results of the reconstruction process applied to four different impoverished objects. Table 6.3 also shows the details for those results, where $\vec{L}$ is a source vector. `Level` shows a degree of being impoverished, `Prior` represents a degree of the prior knowledge of the original objects, and finally `Iteration` is for the number of iterations accroding to each reconstructed model.

	john80	stephane80	hand80	instep80
$\vec{L}(x, y, z)$	(.4, .21, .89)	(.4, .21, .89)	(.4, .21, .89)	(-.17, .03, .98)
Level	80 %	80 %	80 %	80 %
Prior	20 %	20 %	20 %	20 %
Iteration	8	7	8	6

Table 6.3: Summaries of the results for *Album1*

3.1.1 Input images and impoverished models

The left-hand side columns in Figure 6.6 are input images. They are synthesised by the *Lambertian* law. The bottom one is illuminated by a source vector $\vec{L}(-0.17, 0.03, 0.98)$ and the others by $\vec{L}(0.4, 0.21, 0.89)$.

The middle columns are impoverished models. They are 80% impoverished from their own originals. We call them *john80*, *stephane80*, *hand80*, and *instep80* respectively. They preserve the global topology well, but not locally. They have the outlines of the originals in the input images, but the surface details are very impoverished. They preserve the 20% prior knowledge of each original.

The *john80* and *stephand80* are hardly recognisable, because the nose, mouse, and eyes are greatly impoverished. The wrist of the *hand80* is not smooth at all and the toes of the *instep80* have disappeared.

3.1.2 Reconstructed objects

The reconstructed objects shown on the right columns in Figure 6.6 are not optimum from a standpoint of the error estimators. Their *AVID*s and *SDID*s

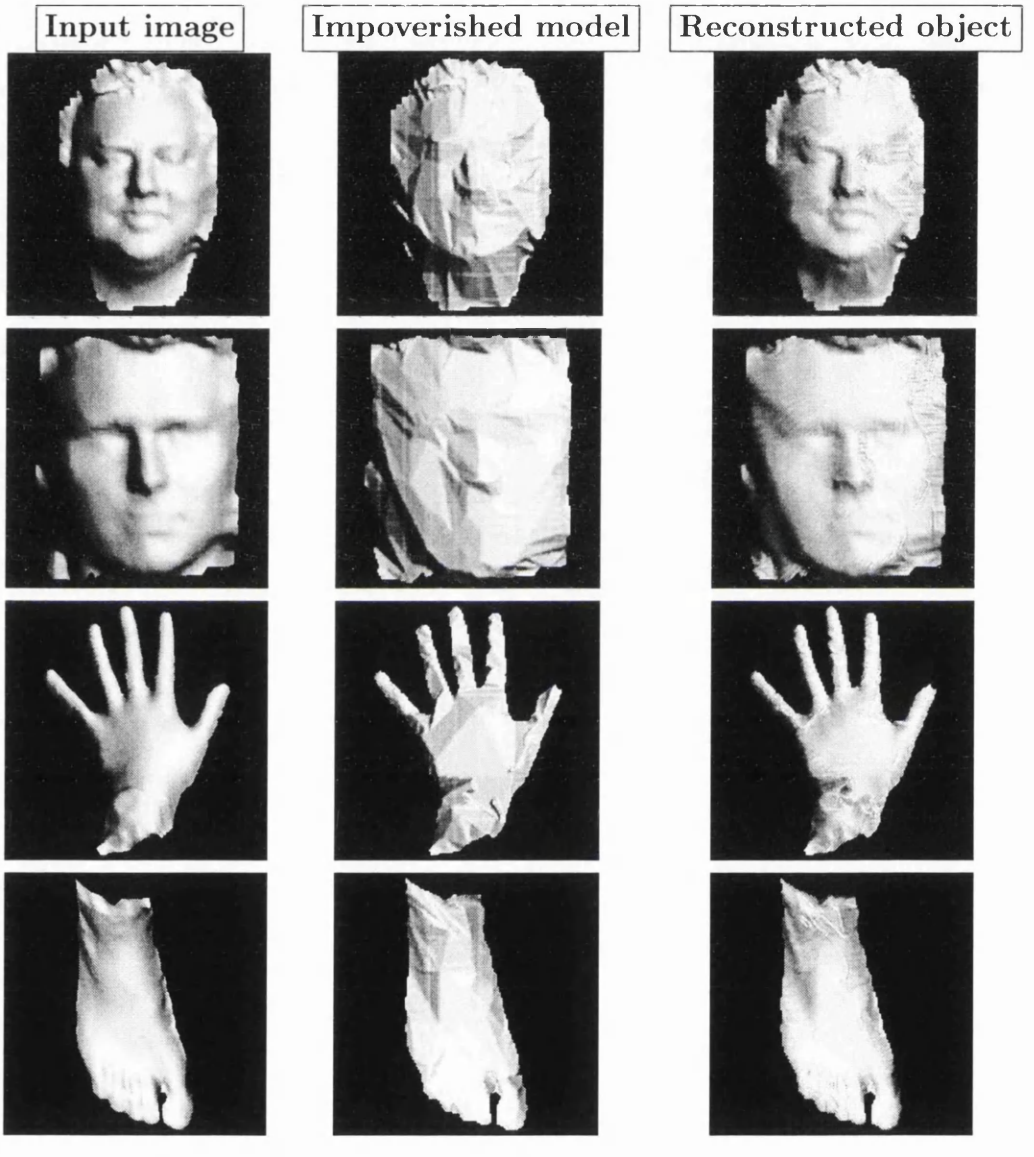


Figure 6.6: Reconstruction album 1

are continuously decreased in 20 iterations, as shown in Figure E.1 and E.2 in *Appendix E.*

The examples were chosen within 10 iterations of the reconstruction. The top column shows a reconstructed face for the *john80* at the 8^{th} iteration. The second top is *stephane80* at the 7^{th} iteration. The *hand80* and *instep80* provide reconstructed objects at the 8^{th} and 6^{th} iteration. Although they have some undesired furrows, for example, which can be found on the right part of the *stephane80*, it should be stressed that they are recognisable.

Figure 6.7 presents a sequence of reconstructed faces from the *john80* in 8 iterations.

3.1.3 Analysis with averages and standard deviations of intensity differences

Table 6.4 shows the *AVIDs* and *SDIDs* for the results. E^0's and S^0's correspond to the impoverished models as shown in the middle columns of Figure 6.6, and E^k's and S^k's are for the reconstructed objects in the right columns of that, and k represents the number of iterations.

		john80	*stephane80*	*hand80*	*instep80*
AVID	E^0	14.27	30.72	12.59	8.95
	E^k	9.09 (8)	23.58 (7)	8.60 (8)	6.42 (6)
SDID	S^0	26.62	38.74	31.33	24.44
	S^k	18.10 (8)	29.45 (7)	24.04 (8)	18.53 (6)

Table 6.4: *AVIDs* and *SDIDs* for *Album1*

For example, the *stephane80* starts with an average intensity error 30.72 and a standard deviation 38.74. As the process goes on, they dropped rapidly at the beginning, as usual. They then slowly decrease. The reconstructed face on the right is created at the 7^{th} iteration, which has a *AVID* value of 23.58 and a *SDID* of 29.45.

The detailed values and curves of the *AVIDs* and *SDIDs* for this album can be found in *Appendix E.*

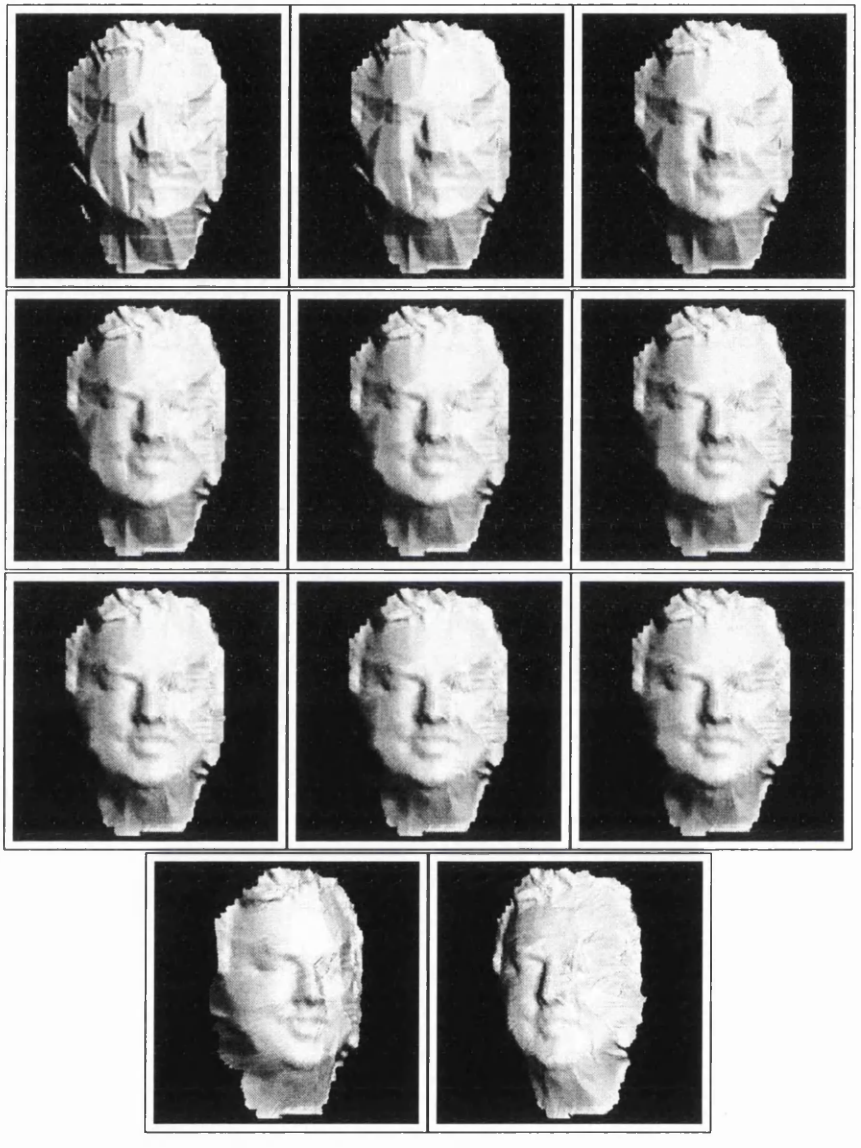


Figure 6.7: A sequence of reconstructed faces from the *john80*

The top left is the 80% impoverished face in Figure 6.6. The first three rows represent the reconstructed faces obtained in 8 iterations from left to right and downwards. The bottom rows show the reconstructed faces at the 8^{th} iteration. They are viewed in different directions.

3.2 Album2 : from a half cylinder and ovoid

Here we demonstrate the possibility of using our reconstruction method with a half-cylinder and a half-ovoid, instead of using an impoverished face. Figure 6.8 and Figure 6.9 illustrate the results starting from a half-cylinder and -ovoid respectively. These are used to reconstruct the three different faces introduced so far in this thesis.

At this point, it should be said that the *AVIDs* and *SDIDs* of the half-ovoid are similar to those of the half-cylinder. To avoid duplicating the discussion, we will follow mainly the half-cylinder here after. The actual values and the curves corresponding to the *AVIDs* and *SDIDs* of the half-ovoid are presented in *Appendix E.*

Table 6.5 presents the results for the half cylinder, where $\vec{L}$ is a source vector and `Level` is the degree of being impoverished. `Prior` shows a degree of the prior knowledge about the original faces in the input face images. Finally `Iteration` is the number of iterations according to each reconstructed face.

	meCL	johnCL	stephaneCL
$\vec{L}(x, y, z)$	(0.40, 0.21, 0.89)	(0.4, 0.21, 0.89)	(0.40, 0.21, 0.89)
Level	N/A	N/A	N/A
Prior	N/A	N/A	N/A
Iteration	14	6	10

Table 6.5: Summaries of the results for *Album2*

3.2.1 Input image and impoverished face

The left-hand side columns in Figure 6.8 and 6.9 are input facial images from top to bottom respectively. They are synthesized by the *Lambertian* law with a source vector $\vec{L}(0.4, 0.21, 0.89)$.

The middle columns are a half cylinder and ovoid used as impoverished faces. They have no information about the originals in the input images, but the images are reconstructed from them. We will call the input images *meCL* (*meOV*), *johnCL* (*johnOV*), and *stephaneCL* (*stephaneOV*) from top to bottom.

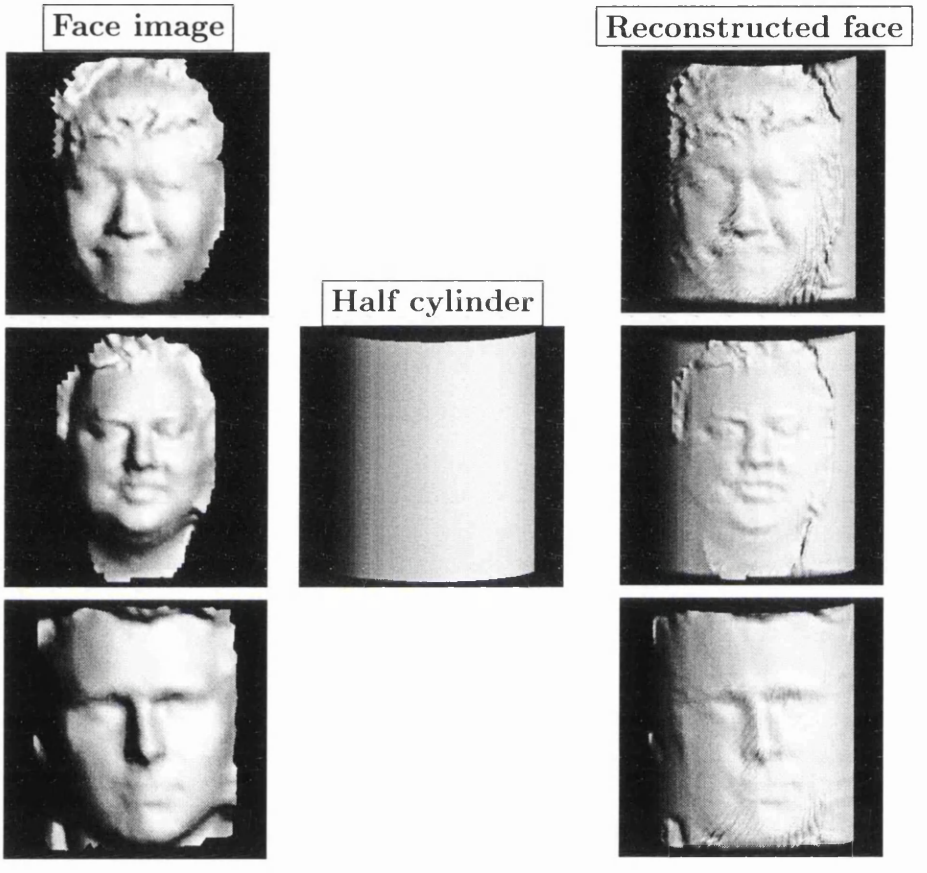


Figure 6.8: Reconstruction album 2

The left-hand side columns show input face images. They are synthesised by the *Lambertian* law with a source vector $\vec{L}(0.4, 0.21, 0.89)$. The middle column is a half-cylinder as an impoverished face. The reconstructed faces, on the right-hand side, are obtained at the 14^{th}, 6^{th}, and 10^{th} iteration from top to bottom.

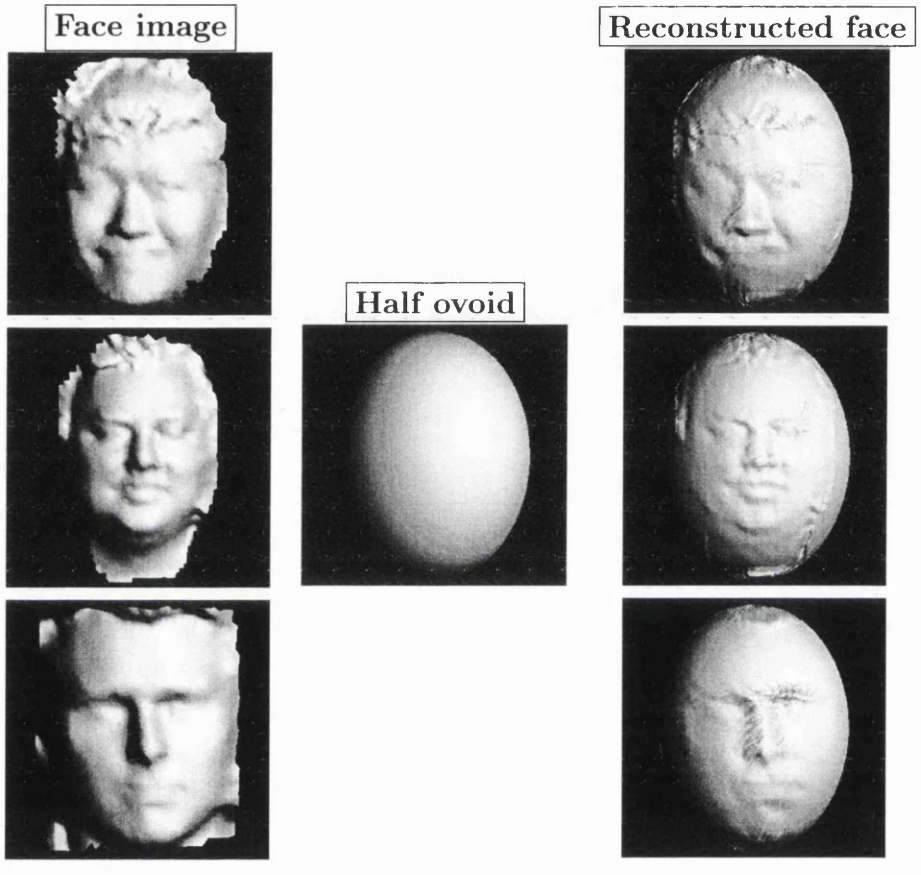


Figure 6.9: Reconstruction examples from a half ovoid

The left-hand side columns show input face images. They are synthesised by the *Lambertian* law with a source vector $\vec{L}(0.4, 0.21, 0.89)$. The middle column is a half ovoid as an impoverished face. The reconstructed faces, on the right-hand side, are obtained at the 10^{th}, 6^{th}, and 10^{th} iteration from top to bottom.

3.2.2 Reconstructed faces

The reconstructed faces shown in the right columns of Figure 6.8 and 6.9 are not optimum. Their *AVIDs* and *SDIDs* are continuously decreased in 20 iterations[7], as shown in Figure E.3 (E.4) and E.5 (E.6) in *Appendix E.* The examples were randomly chosen in those iterations.

The top in Figure 6.8 (6.9) shows a reconstructed face that was created at the 14^{th} (10^{th}) iteration. The middle column was also created at the 6^{th} (6^{th}) iteration. The *stephaneCL* (*stephaneOV*) produced a reconstructed face at the 10^{th} (10^{th}) iteration, as shown at the bottom.

As the iterations go on, the furrows on the results increase. However, the reconstructed faces are nicely carved out of the half cylinder (ovoid) so that they describe well the original faces in their images within a small number of iterations. Unfortunately they look like the half cylinder (ovoid) on the whole, when we see them in the different directions. The bottom rows in Figure 6.10 and 6.11 show the different orientations of the reconstructed faces. This flatness is due to a lack of prior knowledge concerning the original faces. It has a strong connection with the deep-seated problem in the *SFS* methods.

3.2.3 Analysis with averages and standard deviations of intensity differences

Table 6.6 shows the *AVIDs* and *SDIDs* for the results, where average intensity errors E^0's and standard deviations S^0's correspond to the cylinderical face shown at the middle column of Figure 6.8. E^k's and S^k's are for the reconstructed faces in the right columns, and k represents the number of iterations.

		meCL	johnCL	stephaneCL
AVID	E^0	49.44	61.46	38.29
	E^k	38.90 (14)	56.15 (6)	30.59 (10)
SDID	S^0	68.49	74.83	50.98
	S^k	61.10 (14)	71.10 (6)	40.70 (10)

Table 6.6: *AVIDs* and *SDIDs* for *Album2*

For example, the *meCL* starts with an average intensity error of 49.44 and a standard deviation of 68.49, which are maximum. As the iterations progress, the

[7]In face, the *AVIDs* for *meOV* and *stephaneOV* contain each minimum value. This means that the ovoid has more prior knowledge than the cylinder.

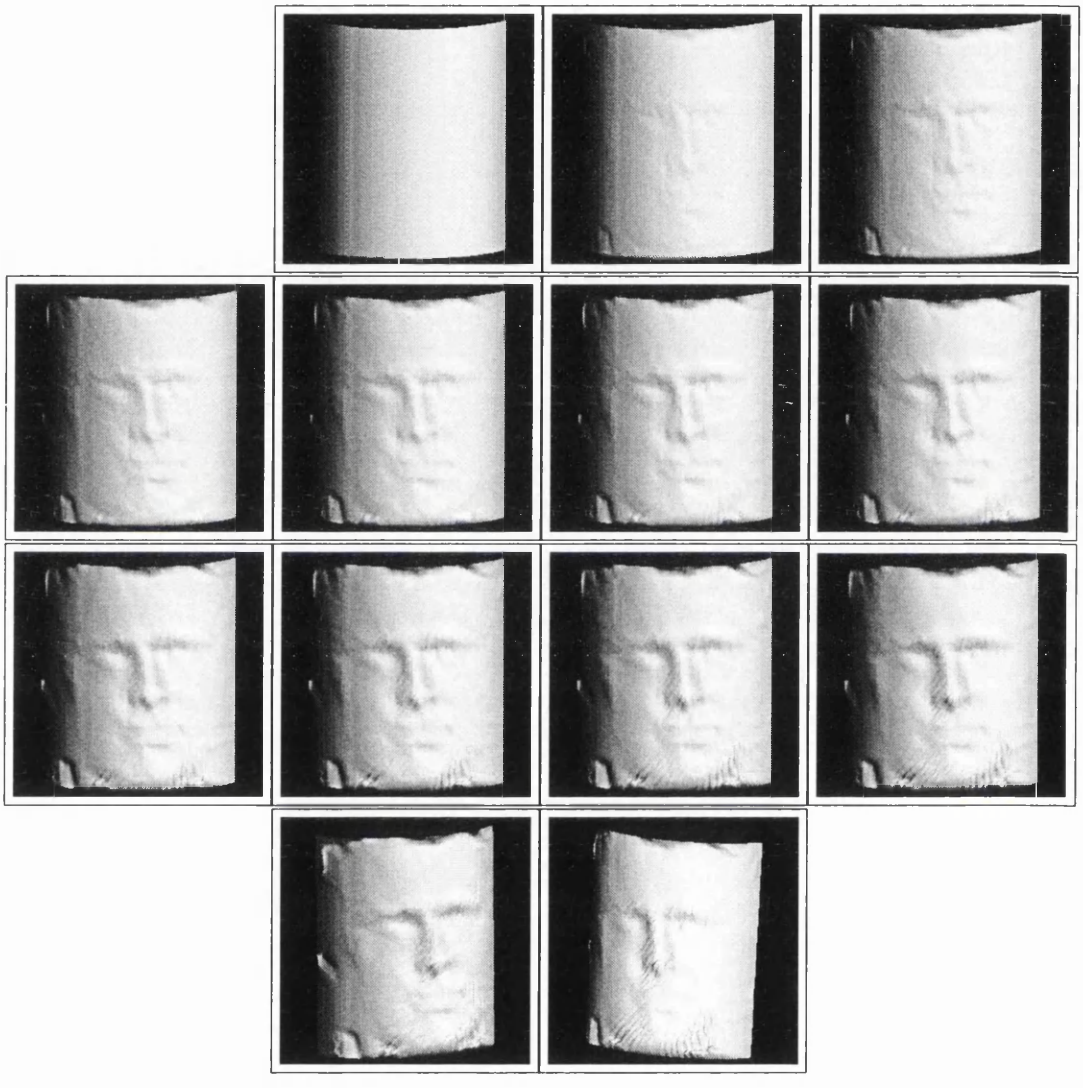


Figure 6.10: A sequence of reconstructed faces from a half-cylinder surface

The top left is the half-cylinder surface as an impoverished face shown in Figure 6.8. The first three rows illustrate a sequence of reconstructed faces obtained in 10 iterations from left to right and downwards. The bottom rows show the reconstructed face at the 10^{th} iteration. It is viewed in the different directions.

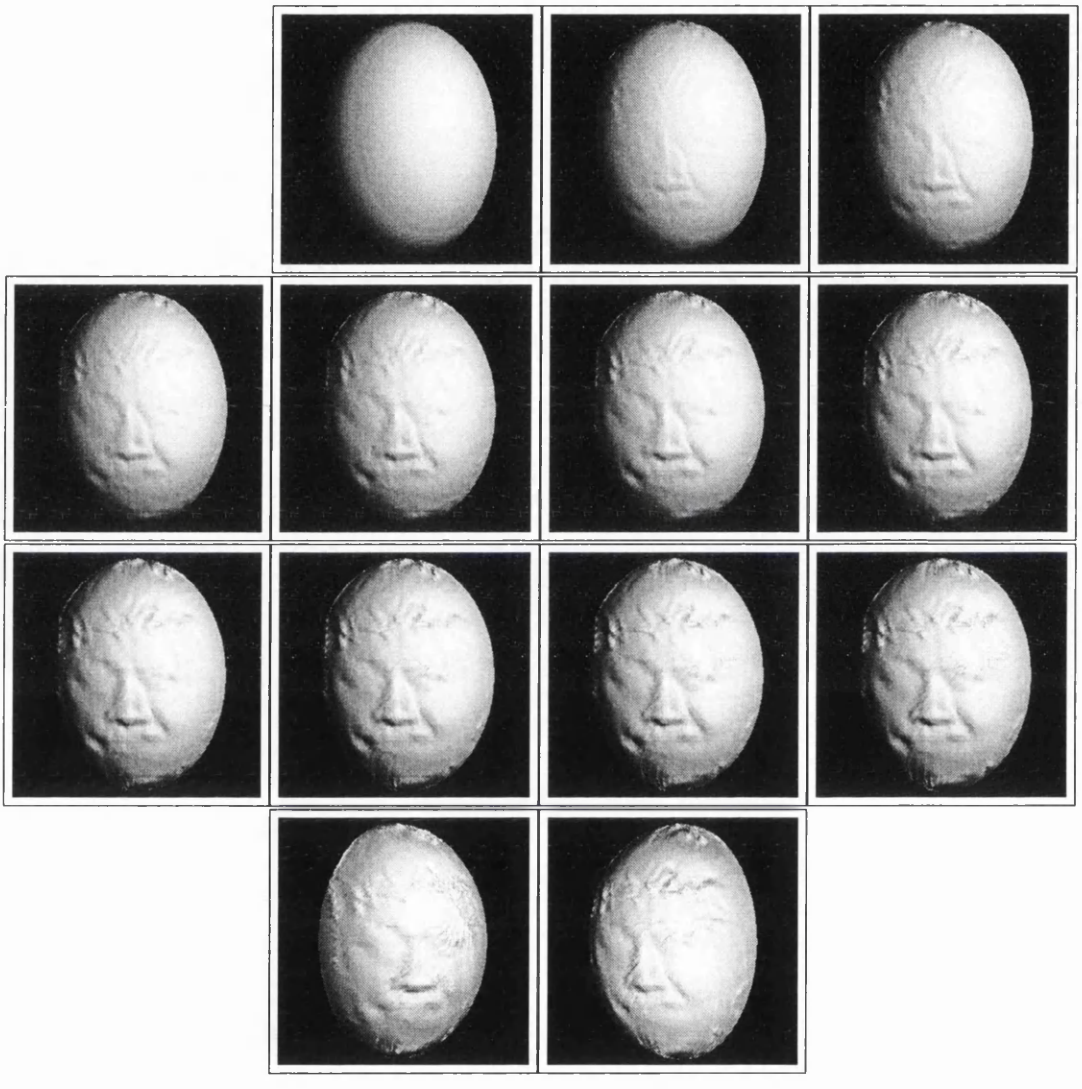


Figure 6.11: A sequence of reconstructed faces from a half-ovoid surface

The top left is the half-ovoid surface as an impoverished face shown in Figure 6.9. The first three rows illustrate a sequence of reconstructed faces obtained in 10 iterations from left to right and downwards. The bottom rows show the reconstructed face at the 10^{th} iteration. It is viewed in the different directions.

error terms drop linearly. A large number of iterations seem to give a good result, however they again produce many furrows. The reconstructed face corresponding to the *meCL* was created at the 14^{th} iteration, which has a *AVID* of 38.90 and a *SDID* of 61.10.

The details for the *AVID*s and *SDID*s of this album can be found in *Appendix E.*

3.3 Album3 : from a flat surface

Here we demonstrate the possibility of using our reconstruction method with respect to using a flat surface as an impoverished face. Figure 6.12 shows the results of using a flat surface to reconstruct four different faces.

Table 6.7 represents the results, where $\vec{L}$ is a source vector and `Level` is a degree of being impoverished. `Prior` shows a degree of the prior knowledge about the original faces in the input face images. Finally `Iteration` is the number of iterations for each reconstructed face.

	Lena	Mozart	mePhoto	yhPhoto
$\vec{L}(x,y,z)$	(.78, .16, .61)	(.50, .50, .70)	(.78, .16, .61)	(.78, .16, .61)
Level	N/A	N/A	N/A	N/A
Prior	N/A	N/A	N/A	N/A
Iteration	6	8	11	8

Table 6.7: Summaries of the results for *Album3*

3.3.1 Input image and impoverished face

In Figure 6.12, the first two columns on the left-hand side are well synthesised by the *Lambertian* shading model[8], called *Lena* and *Mozart*. The *Lena* has a source vector $\vec{L}(0.78, 0.16, 0.61)$. The *Mozart* is illuminated by $\vec{L}(0.50, 0.50, 0.70)$.

The next two columns are photographs of human faces, called *mePhoto* and *yhPhoto*. They were digitised by a *Canon CLC10* scanner, and then formatted by *PGM* to use as the face images in our method [Mv96]. We assume that

[8]They were adapted from the *Computer Science Department, University of Central Florida*, Orlando, USA.

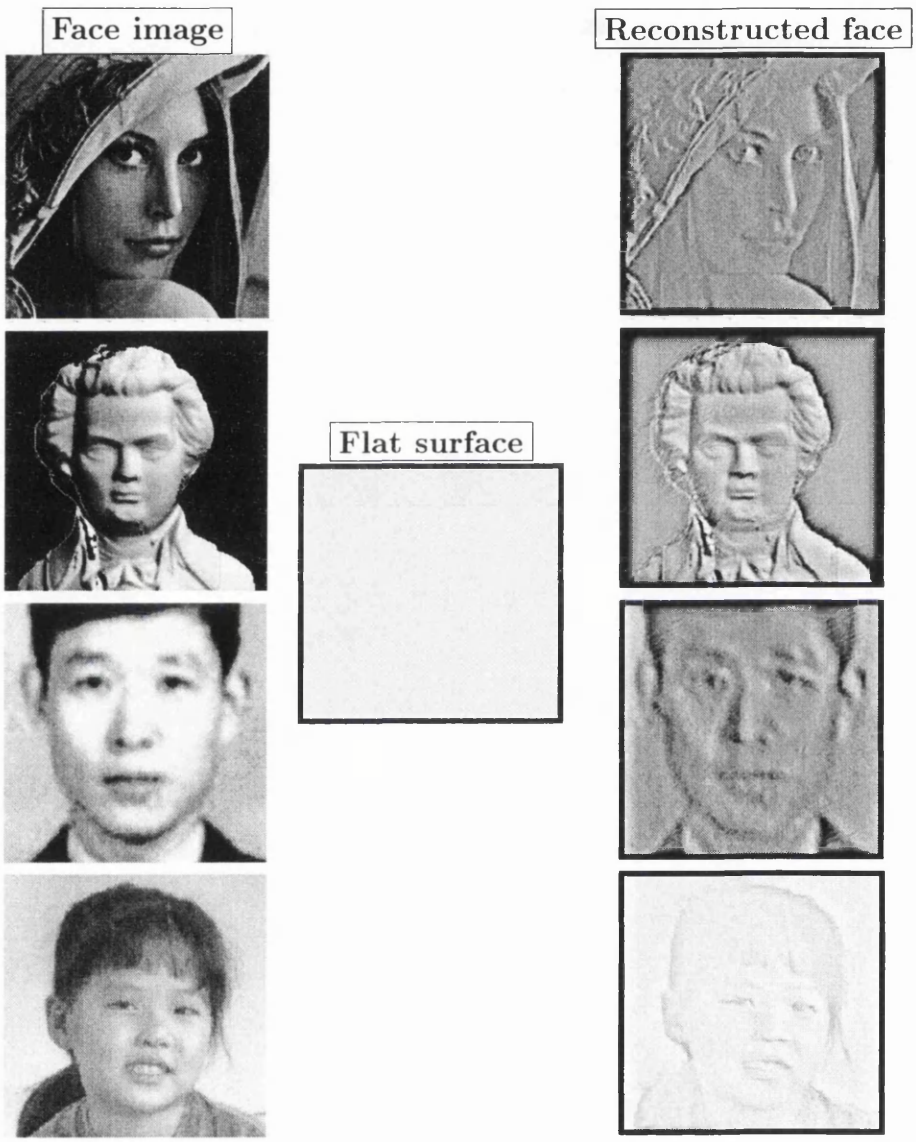


Figure 6.12: Reconstruction album 3

these two photographs were shaded by the *Lambertian* law with a source vector $\vec{L}(0.776, 0.161, 0.609)$. This is the same as that employed for *Lena*.

The middle column is a flat surface used as an impoverished face. It was generated from a uniform grid. They have no information about the original faces in the input images.

3.3.2 Reconstructed faces

The reconstructed faces shown in the right columns of Figure 6.12 are not optimum from a standpoint of the error estimators; that is, since the error values are decreased continuously in 20 iterations, as shown in Figure E.7 in *Appendix E*, they were chosen at random. They were created at the 6^{th}, 8^{th}, 11^{th}, and 8^{th} iteration from top to bottom respectively. The reconstructed faces resemble the original faces in their images within a small number of iterations[9].

However they suffer from the same flatness problem as conventional *SFS* methods. Figure 6.13 shows a sequence of reconstructed faces for the *yhPhoto* in 8 iterations. The bottom rows are the reconstructed face at 8^{th} iteration. It is viewed in the different directions, but it still looks like images. It seems probable that this flatness is also due to a lack of prior knowledge as the reconstruction from the half-cylinder and -ovoid in the *Album2*.

We will discuss in more detail one possible way to avoid the flatness in reconstructing an original face, starting from a half-cylinder or a flat surface, in the future study section of the final chapter.

3.3.3 Analysis with averages and standard deviations of intensity differences

Table 6.8 shows the *AVID*s and *SDID*s for the results, where average intensity errors E^0's and standard deviations S^0's correspond to the flat face as shown in the middle column of Figure 6.12. E^k's and S^k's are for the reconstructed faces in the right columns, and k represents the number of iterations.

For example, the *Mozart* starts with an average intensity error of 105.36 and a standard deviation 73.32, which are maximum. As the number of iterations

[9]It should be said that the result of the top in Figure 6.12 is distinguishable from that in Figure 2.11 in *Chapter 2*. That is, the former was obtained at the 6^{th} iteration, while the latter was obtained at the 2000^{th} one. In addition, the quality of the former is clearer than that of the latter. Finally, the running time corresponding to the latter is around 240 seconds as shown in Table 2.1, but ours is around 60 seconds, assuming that they are in the same condition.

Figure 6.13: A sequence of reconstructed faces from a flat surface

The top left is the flat surface as an impoverished face shown in Figure 6.12. The first three rows illustrate a sequence of reconstructed faces generated in 8 iterations from left to right and downwards. The bottom rows show the reconstructed face at the 8^{th} iteration. It is viewed in the different directions.

		Lena	Mozart	mePhoto	yhPhoto
AVID	E^0	78.57	105.36	60.96	59.49
	E^k	71.49 (6)	85.77 (8)	51.17 (11)	52.21 (8)
SDID	S^0	46.59	73.32	46.52	51.12
	S^k	43.71 (6)	67.75 (8)	46.07 (11)	50.61 (8)

Table 6.8: *AVID*s and *SDID*s for *Album3*

progresses, they drop linearly. So a large number of iterations seems to give a good result. However, they again produce many furrows. The reconstructed face corresponding to the *Mozart* is created at the 8^{th} iteration, which has a *AVID* of 85.77 and a *SDID* of 67.75.

The details for the *AVID*s and *SDID*s of this album can be found in *Appendix E* as well.

4 Discussion

In this chapter, we defined two statistical estimators to analyse the intensity error terms with respect to reconstructed faces.

One is an arithmetic average of intensity differences between two images *Oimage* and *Rimage*, the so-called *AVID*. It represents an intensity error of a reconstructed face at the k^{th} iteration. The other is a standard deviation of the intensity differences from an *AVID*, called *SDID*. It represents a degree of dispersion of intensity errors for a reconstructed face.

We conducted a number of experiments on our reconstruction method. They can be grouped into four with respect to their impoverished faces: one is the differently impoverished faces, another is the different objects, the third is the half-cylinder and ovoid, and finally the flat surface.

When we apply under 80% impoverished faces to our method, empirically, it produces the reconstructed faces with minimum error terms in 20 iterations; for example, the *me20*, *me40*, *me60*, and *me80*. For the over 80% impoverished faces, the average intensity errors and the corresponding standard deviations of reconstructed faces continuously fall in those iterations, except the *meOV* as shown in Figure E.4 and E.6. It is clear from the experiments that our method is stable. So, the reconstructed faces converge into the originals in that period.

However, our method has a drawback; it strongly depends on the impover-

ished faces given at the initial stage. For example, if an impoverished face has much of the information in an original, as with *me20* in Figure 6.2, our method can give an optimally reconstructed face. On the contrary, if the information of the original face is sparse; for example, a cylinder or flat surface, our method suffers from the same flatness problem as conventional *SFS* solutions. To overcome this problem, we will suggest a possible way forward in the final chapter.

Another drawback of our method is the production of furrows on reconstructed faces, even though the average intensity errors corresponding to them decrease as the iteration goes on. We believe that the *moving-vertices* operation and bad model-normals result in those furrows. The final chapter also presents a possible means of avoiding this problem.

Chapter 7

Conclusion and Future Work

This final chapter summarises what has been achieved in this thesis. It also presents the directions in which its techniques may be developed in the future.

1 What Has Been Achieved

Initially, the central problem of this thesis was to develop a method for reconstructing 3D models from single face images that only contain 2D information in the form of variations in intensity (shading). Unfortunately the conventional *SFS* methods suffer from a number of problems including the flatness of reconstructed surfaces.

To overcome the flatness problem, we employed a geometrical model, the so-called impoverished face to provide prior knowledge. A reconstructed face created by the new method is a revision of an impoverished face to reflect exactly an approximation of the original face in an image as argued in *Chapters 5* and *6*.

Consequently, a reconstructed face looks correct even when it is turned to a different orientation compared with the one in the input image. Figure 7.1 shows the reconstructed face presented in Figure 5.1 in *Chapter 5*. It is differently viewed, together with an impoverished face. The upper rows are the $90^{+}\%$ impoverished face before applying our reconstruction method. The left columns are viewed in the right-hand side direction (20^{o}) with a source vector $\vec{L}(0.40, 0.21, 0.89)$, while the right columns are viewed in the left-hand side direction as much with a different source vector $\vec{L}(0.15, 0.08, 0.98)$. The middle ones are viewed in the frontal direction with a source vector $\vec{L}(0.40, 0.21, 0.89)$.

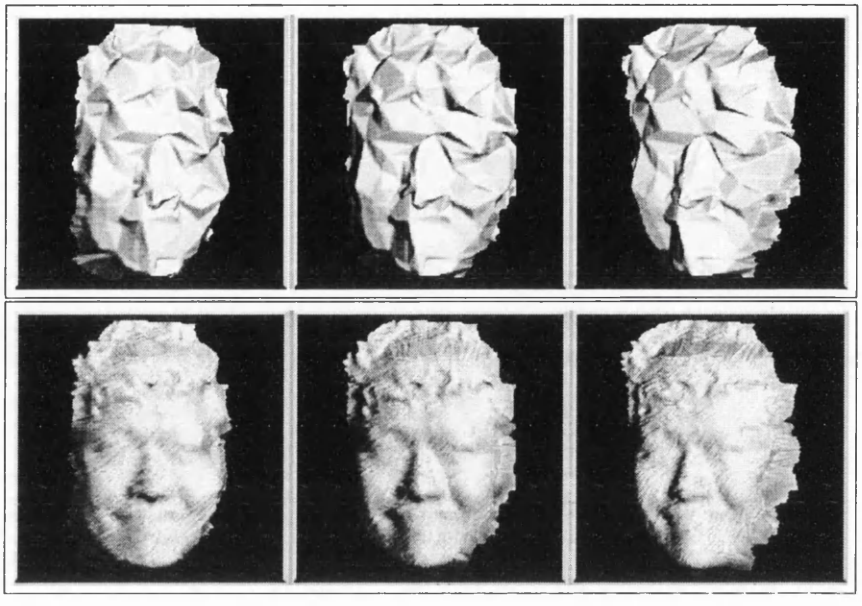


Figure 7.1: An exapmle of a reconstructed face with different views

The left columns are viewed in the right-hand side direction (20^o) with a source vector $\vec{L}(0.40, 0.21, 0.89)$, while the right columns are viewed in the left-hand direction as much with a different source vector $\vec{L}(0.15, 0.08, 0.98)$. The middle ones are frontal views with a source vector $\vec{L}(0.40, 0.21, 0.89)$.

1.1 Revisiting overall reconstruction procedure

The conceptual task of reconstructing an approximation of the original face in an input image, starting from an impoverished face, is decomposed into three sub-problems of: representing prior knowledge (*Chapter 3*), deriving a shape from prior knowledge (*Chapter 4*), and face reconstruction (*Chapter 5*). These problems can be integrated in a way shown in Figure 7.2.

The discussion in *Chapter 3* was focused on how we represent prior knowledge for human faces and how much prior knowledge is sufficient for our method. To answer them, we represented prior knowledge using an impoverished face, because it contains the prior knowledge of the original face to some degree. So, it can be considered as a coarsely sculptured face or a burn face.

To obtain an impoverished face in practice, we have developed two tools: *Meducer* and *Meditor*. The goal of the Meducer is to impoverish original faces

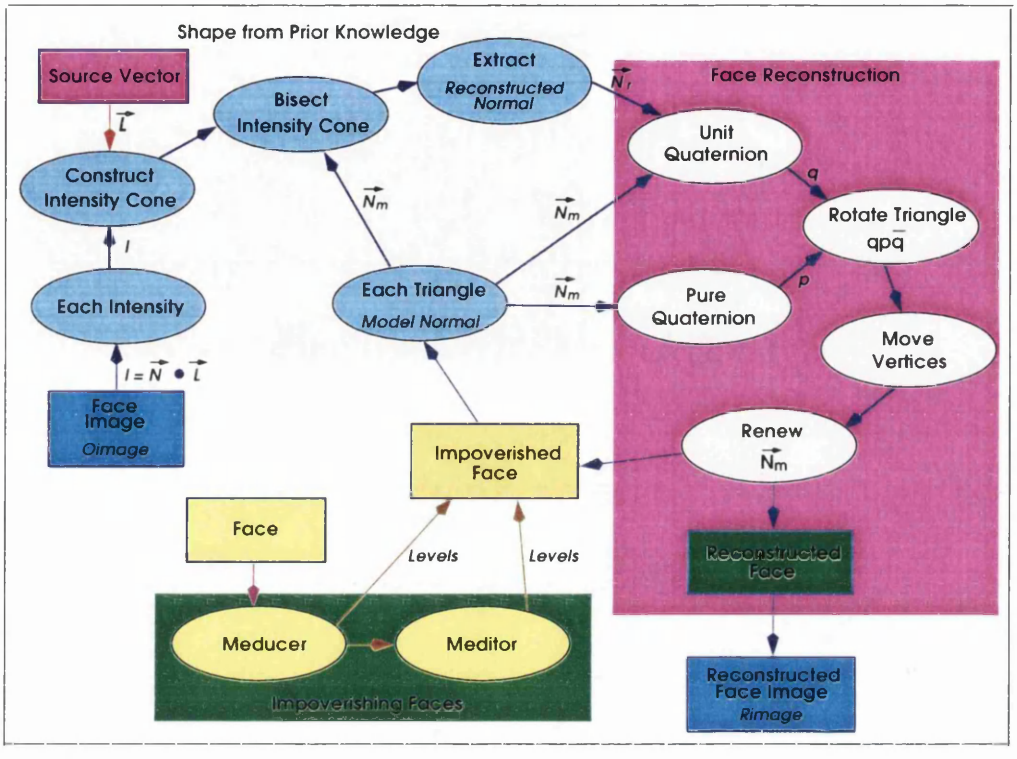


Figure 7.2: Main stream of overall face reconstruction

in terms of their triangles as much as required. It is based on the removal of a geometric vertex and the retriangulation of a ring which is a closed path along with adjacent vertices to the removed vertex. One of three operations is applied to retriangle the ring, which are the high, low, and neighbourhood operation.

The Meditor provides an interactive way of correcting the undesired triangles created in impoverishing a face in the form of a mesh. It is endowed with five operations: vertex deleting, vertex adding, triangle adding, edge swapping and edge deleting.

In *Chapter 4*, we reformulated the conventional *SFS* problem into the problem of deriving shape from prior-knowledge. For a given source vector, an intensity value at an image point constructs an intensity cone which is formed by the candidates of a surface normal. By means of bisecting the cone, a model normal from a triangle corresponding to the image point cuts the infinite number of candidates down to two. The candidate that is closer to the model normal becomes the reconstructed normal as a unique solution.

In *Chapter 5*, we discussed face reconstruction in more detail. We start by defining two quaternions which provide a means of rotating a triangle speci-

fied by a model normal into another specified by a reconstructed normal. The vertices of the rotated triangle are moved into their new positions to avoid the problems created by the rotation. After that, the model normals of the triangles sharing the moved vertices are renewed. The face reconstruction procedure was demonstrated on a very impoverished face.

1.2 Assessment of results

To make a reconstructed face acceptable, the face reconstruction procedure is restarted with a new impoverished face in an iterative fashion. In *Chapter 6*, we defined two statistical estimators to analyse a face reconstructed iteration by iteration.

One is an arithmetic average of intensity differences between two images *Oimage* and *Rimage*, which is called an *AVID*, and the other is a standard deviation of intensity differences from the *AVID*, called a *SDID*.

We used these measures in a number of experiments. These investigated of variously impoverished faces, different objects, a half cylinder (ovoid), and a flat surface. Empirically, we found that the *AVID* and *SDID* values with respect to the number of iterations depend on an impoverished level of a face.

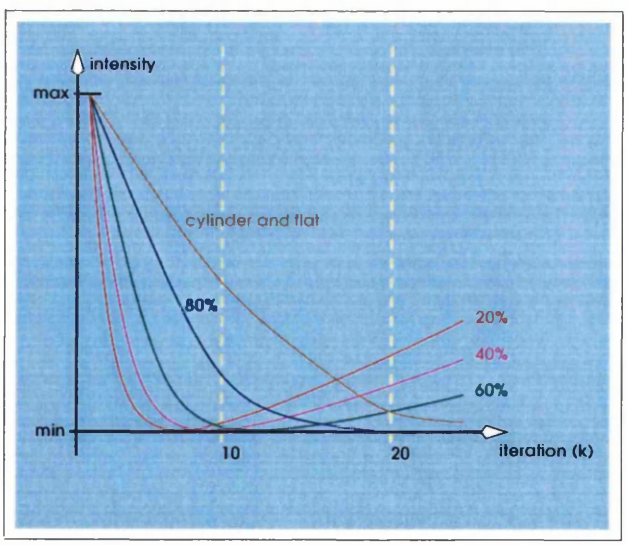


Figure 7.3: General shapes of error estimators according to impoverished levels

Values of error estimators in terms of iterations depends on impoverished levels of a face.

Figure 7.3 shows the collective shapes plotted by those values. As an impoverished level goes higher, the shape plotted by them is more stretched. The levels impoverished as much as 20%, 40%, and 60% can create the reconstructed faces with each minimum error term in 10 iterations.

An 80% or more impoverished level produces a shape that drops dynamically

at the beginning and then slowly approaches a constant value as a minima. On the other hand, a cylindrical or flat face creates a shape that decreases linearly as each iteration progresses.

From the experiments conducted, it is reasonable to believe that our reconstruction procedure can successfully produce an acceptable approximation of the original face in a single image. Usually a reconstructed face obtained by around 10 iterations is `recognisable` as the original face in the image.

At this point, we should answer the question that how much information (prior knowledge) about an original face is sufficient to reconstruct an `acceptable`[1] face from an impoverished face.

In practice, it is an uneasy question to answer correctly, because the impoverishment degree highly depends upon the 3D appearance of an individual. For example, although both *john80* and *stephane80* in Figure 6.6 were impoverished as much as 80%. Strictly speaking, they are not same degree each other, because their sizes are different from each other. To avoid this difficulty, all original faces and images should be carefully prepared by means of standardised parameters including face size, features, and their positions and sizes, and so on.

However, from the result in Figure 6.2, we can say empirically that around 20% prior knowledge is required to reconstruct an acceptable face, in the sense that a reconstructed face does not suffer from the flatness.

2 Future Directions

The scope for future work in this thesis can be categorised into two broad classes:

- Topics that improve the current work directly addressed in this thesis.
- Potential applications for face reconstruction from real photographs.

[1]The term `acceptable` is different from `recognisable`. The former is focused on the 3D appearance of a reconstructed face, but the latter is not. For example, the reconstructed face in Figure 6.13 is not acceptable but recognisable, because it still looks a 2D image even if it is viewed in the different directions. However, the reconstructed face in Figure 7.1 is acceptable.

2.1 Some possible improvements

2.1.1 Meditor

We have demonstrated the possibility of using a half-cylinder (-ovoid) and a flat surface as an impoverished face in *Chapter 6* when information about an original face is not available. The reconstructed faces are recognisable as the original faces in input images. However they are flattened as the features on a coin are.

This problem can be addressed by extending the five basic operations of the *Meditor*. If we extend it to edit the features in terms of the relative depth such as *3D paint* [Wil90] and *FACES* [PW91], we can apply our reconstruction procedure to the edited face. We believe that the original face in a photograph may be reconstructed using the *Meditor*.

2.1.2 Minimising changes

A shortcoming of our method is that small furrows are created on the reconstructed faces. We believe these are due to moving vertices, as mentioned in *Chapter 5*. Here we briefly suggest a way of avoiding these furrows in future work. To make the discussion simple, we reuse everything shown in Figure 5.5 and the impoverished face in Figure 5.6 in *Chapter 5*.

Figure 7.4 shows each stage in the process of face reconstruction using the minimising-changes operation. The basic idea is, after rotating a triangle, that the heights of all other triangles in the impoverished face are adjusted in order to minimise the geometric changes. In *stage a*, after the rotation of T_1, the heights of all the other triangles T_2, T_3, and T_4 are adjusted to keep their orientations unchanged as before. The vertical arrows on vertices in Figure 7.4 represent the height adjustment.

On the other hand, the dashed lines represent the triangles as before and the shaded circles show the boundary of a rotation of a triangle. In *stage b*, the rotation of T_2 adjusts the other triangles T_1, T_3, and T_4 in a similar fashion. If we apply the rotation and the height adjustment to the triangles T_3 and T_4 as well, we can obtain the reconstructed face converging to the approximated mesh as shown in Figure 5.5, in only one iteration.

This method seems to be promising geometrically; however, it may consume a lot of time to adjust the height fields of all other vertices whenever a triangle rotation occurs. Therefore we must consider speedups in implementing the *minimising-changes* operation.

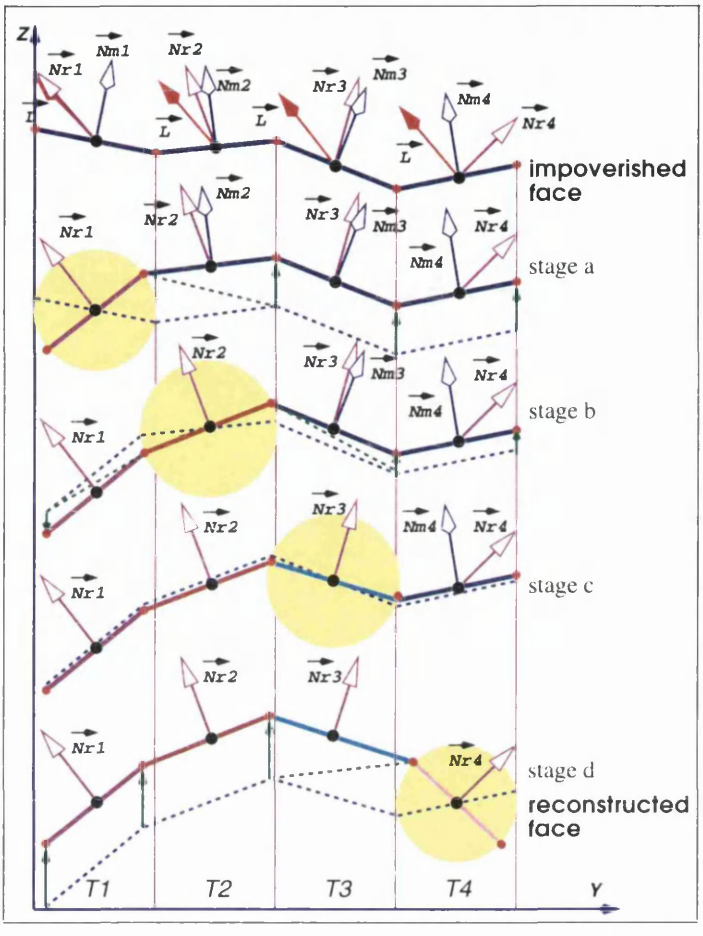


Figure 7.4: Minimising-changes operation

After rotating a triangle, the heights of all other triangles are adjusted in order to minimise the geometric changes. Dashed lines represent the triangles as before and Shaded circles show the tracks of rotations. In addition, a vertical arrow on each vertex shows the height adjustment. The reconstructed face shown on the bottom accurately converges to the approximated mesh shown in Figure 5.5.

2.1.3 Generalised face model

The aim of this thesis is to recover 3D facial appearances from an image, with a geometrically impoverished model, as if a sculptor carves a coarsely sculptured stone into a face holding a face photograph in his hand. The impoverished model may be derived from a mean face out of a number of 3D scanned faces [Muk95, AGR95, OVBT95] or a generic face model widely used in the model-based coding techniques [ASW93].

The state of the art in this kind of image analysis initially requires us to adjust a face model so that the structure of the model corresponds with the facial features of an image to any reasonable degree of accuracy. This adjustment could be accomplished by the geometric transformations including changes in orientation, size, and shape of the model.

2.1.4 Finding the source vector in an image

Finding the direction of a source vector turns out to be a major problem in reconstructing the original face from an ordinary photographic image. Fortunately there are many successful algorithms for doing so including [Pen82, BH85, LR85, ZC91]. Adopting one of them, we can apply our method to any facial photographs.

2.1.5 Realistic rendering

In this thesis, a reconstructed face in the form of polygonal mesh is shaded by a *Phong-like* rendering method mentioned in *Chapter 2.* In terms of realism, the shortcomings of the shaded face are that it has a rather synthetic visual quality. This is because the shading models calculate the intensity on a surface point which is linearly interpolated from a polygon for a known light source. The shading models are unable to portray the details of facial features such as wrinkles, skin texture, moles, freckles, pupils, eyebrows, mustache, and so on.

Texture mapping is generally applied for enhancing the realism of a face. Even when the geometry of the face is coarse more or less, a texture mapped face looks like a real photograph [BN76, Hec86, YD88]. So, if we employ a texture-mapping algorithm, a reconstructed face may look better.

2.2 Potential applications

The field of 3D reconstruction is likely to grow significantly in the next few decades.

2.2.1 Meducer

The *Meducer* as a tool providing an impoverished face is not accurate in a very low level. For example, it locally creates undesired triangles on an 80% impoverished face, as mentioned in *Chapter 3.* This often requires us to edit the face. However, the *Meducer* is sufficient for our purpose in this thesis, because we need an impoverished face as a coarsely sculptured face or a burn face. In our experiments, we used impoverished faces without removing the undesired triangles.

The *Meducer* globally impoverishes a face in the form of a triangulated mesh. If we can describe a fire victim's face using impoverishment, it may simulate a

burn without a real face.

2.2.2 Reconstructing a burn face

If we could achieve the simulation on burn faces accurately, it seems probable that we reconstruct the original face from a pre-burn photograph and a burn face. Then reconstructed faces can be used to design individual masks so that they are used to simulate a plastic operation for facial burns. These masks are currently designed using laser scans of the person's face which are expensive and not always feasible for burn victims [AGR95].

Our method is likely to be suitable for this purpose. Again, a simulated-burn face can act as an impoverished face.

Another possibility is to improve the 3D reconstruction of images extracted from movie sequences where insufficient landmark data is otherwise available. That is to say, if the landmark data could be represented as a coarse face mesh, we may apply our method to the coarse mesh so as to refine it.

2.2.3 Reconstructing a missing child's face

Figure 6.12 in *Chapter 6* shows an example of reconstructing the original face in a photograph, using a flattened face, when a geometrical impoverished face is not available. This implies that we could reconstruct a missing child's face from a photograph. However the reconstructed face appears like bas-relief on a coin. If the *Meditor* allows a coin-like face to be improved to look more like a real one, our method may reconstruct the original face more accurately.

To extract 3D shape from only raw photographic data more accurately, we should consider both analysing digitised photographs and obtaining an impoverished face instead of using a flattened face. Real digitised photographs usually contain specular reflectance components. These components had not been analysed carefully in the thesis, because one of the main thesis objectives was to employ geometrical prior knowledge of a face in order to overcome the flatness problem in *SFS* techniques.

2.2.4 Reconstructing a cyber-face

There are some techniques to merge two or more face images to produce one 2D composite image [Gal78, BP92a]. A similar effect can be observed when we see two different face images in a stereoscopic way.

If we choose a different person's face model instead of using an impoverished face, we could obtain an interesting result; a composite 3D face that is a cyberface from one face of the model to the other of an input image.

Appendix A

Mesh Simplification Methods

In *Chapter 3*, we have discussed the *Meducer*, which is a tool for impoverishing a face in the form of a polygonal mesh. One reason for doing this is to provide our reconstruction method with a representation of prior knowledge about the original face.

There are many possible ways to impoverish a polygonal mesh, called `mesh simplification`. They are usually used for representing a minimum approximation of a physical subject from sampled 3D points. These points are provided by the special hardware equipments; for example, a laser scanner, a computed tomography, and so on [SBD86, WT91, SZL92, Tur92, Hop94, HDD+93, Var94, AS96, RR96]. So, it is interesting to survey those methods.

In this appendix, we will discuss the issues of mesh simplification. They may be connected with the face model required in our investigation to some extent: what is mesh representation?, what is mesh simplification and why is it useful?, and finally the classification of various simplification methods.

1 Mesh Representation

A polygonal mesh consists of a set of vertices, edges, and polygons. A vertex is shared by at least two edges. An edge connects two vertices and is part of at least one polygon. A polygon is a closed sequence of vertices specified in either clockwise or anticlockwise order, or equivalently a closed path of edges.

For example, Figure 3.5 in *Chapter 3* shows an example of a polygonal mesh, which consists of a set of 6 vertices, 10 edges, and 5 polygons. A polygon is a closed sequence of three vertices in anticlockwise.

On the other hand, a polygonal mesh can be represented in different ways depending on the purpose. A vertex-based representation is commonly used in surface reconstruction [WW93, RR96]. An edge-based representation is mainly used in real-time rendering methods [Woo85, FvDFH90]. In the thesis, we employed a mixed representation. That is, we represent a face in both ways, as shown in Figure 3.4 in *Chapter 3.*

2 Mesh Simplification and Usefulness

Mesh simplification is a transformation of reducing the number of polygons in an original mesh. It enables the rendering of the fewest number of polygons to represent an object while minimising the perturbation of the original shape.

Mesh simplification is useful in rendering a distant object or in reducing the geometric redundancies of an object. As an example of rendering, we assume that a graphics application renders a distant sphere consisting of $10,000$ polygons. Since the sphere is far away, it requires only a few pixels in the image. In this case, it is unnecessary to use the fully detailed mesh. Instead, say, a 100 polygons simplified may be enough, and yet the resulting image looks approximately same as the original.

As another example, we may imagine that each side of a cube consists of $1,000$ polygons. In this case, since all polygons in one side have the same orientation, they are redundant. To avoid the orientation redundancy, they can be substituted for only one polygon, because it is enough to represent for their orientations.

3 Mesh Simplification Methods

There are many uniquely proposed algorithms to simplify a given mesh thus far. Amongst them, it is difficult to find one general purpose technique that simplifies any given mesh, because there is a wide range of objects that can be represented by polygonal meshes.

For instance, techniques that are successful in a mesh of a building, which is a collection of block shapes, may not be applicable to a mesh of human face, which is a set of smooth surface patches. This is one reason why there are many algorithms.

These simplification algorithms can collectively be classified with three groups

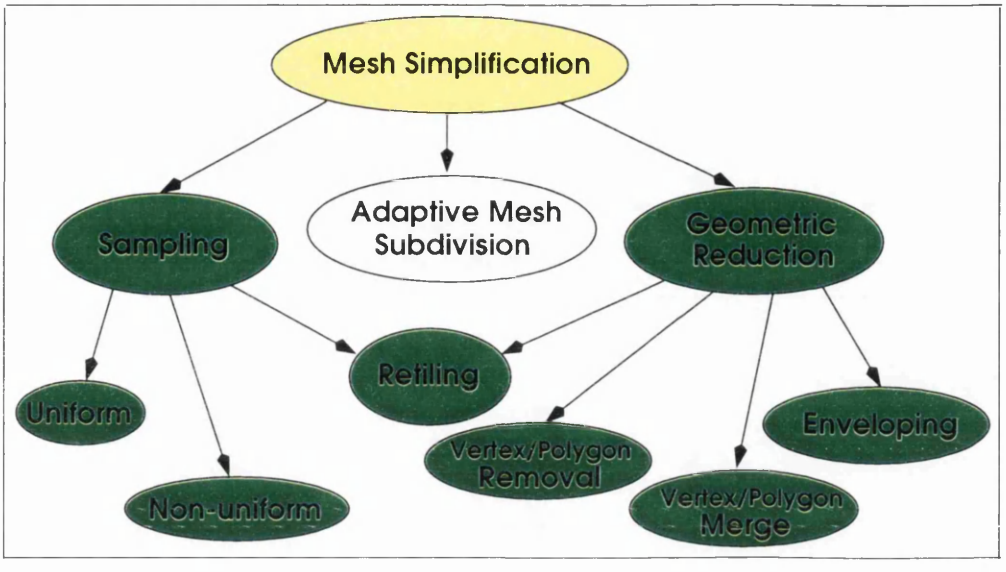


Figure A.1: Mesh simplification tree

Mesh simplification methods are classified with three aspects: Sampling, adaptive mesh subdivision, and geometric reduction. In our investigation, the uniform sampling and vertex removal method have been developed to simplify a human-face model.

as shown in Figure A.1 [Eri96]; a sampling, adaptive mesh subdivision, and geometric reduction group. The sampling group can be subdivided into two algorithms according to their sampling fashion. In addition, the geometric reduction group can be subdivided into three algorithms with respect to the subject of reduction. At this point, it should be said that the retiling algorithm emplys both the sampling and geometric reduction aspect. Each group will be separately discussed more in detail later.

Now a question is raised; what is a criterion of simplification? There are two bases to guide simplification algorithms; a local or a global criterion. The former is a measure of how much the shape changes locally [SZL92, Tur92, Var94], the latter is a measure of simplification error considered as global shape parameter [Hop94, RR96].

3.1 Sampling

Sampling group simplifies an original mesh by taking some of its vertices in either uniform or non-uniform way. Figure A.2 shows an example of each way. The left-hand side is an original mesh in the form of rectangles. The middle is a simplified mesh consisting of the vertices sampled uniformly. In addition, the right-hand side is obtained by sampling vertices according to local surface curvature; the vertices in the mountain shape area were sampled densely, while those in the plain shape area were sampled sparsely.

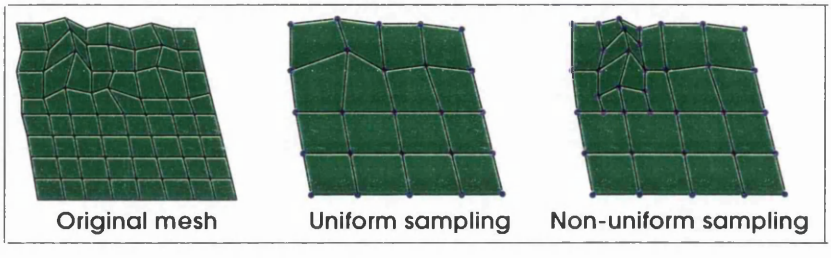


Figure A.2: An example of mesh simplification using Sampling

> Sampling group simplifies the original mesh by taking surface points in either uniform or non-uniform way. The vertices marked are the points sampled in the figure.

The level of simplification is determined by specifying the resolution of overlaying grid or the number of sampled vertices. The uniform sampling is suitable for a smooth surface because the variation of its geometry is not much. However, the non-uniform sampling is suited to a varied surface such as a mountain. In the latter case, it is necessary to keep ridge and valley vertices so as to minimise the perturbation of the original shape.

On the other hand, the retiling algorithm simplifies the original mesh by resampling the vertices, which are not a subset of those in the original, and by removing original vertices [Tur92, HDD+93]. It works better on smooth surfaces rather than sharp ones.

3.2 Adaptive mesh subdivision

Adaptive mesh subdivision algorithm simplifies an original mesh by adding vertices locally, starting with a simple base mesh [SBD86, MJDZ91, WT91, KT96]. Figure A.3 illustrates an example of this algorithm.

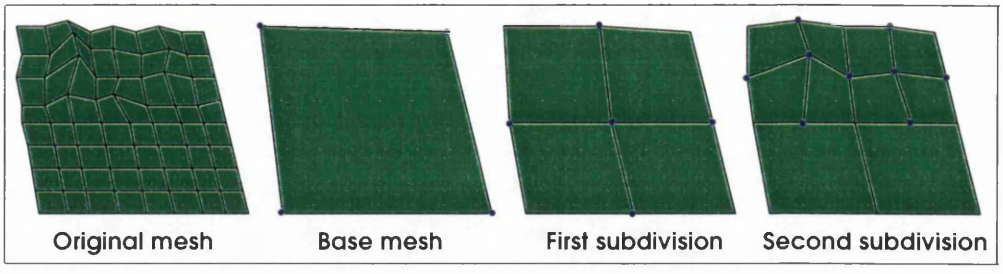


Figure A.3: An example of mesh simplification using adaptive mesh subdivision

Adaptive mesh subdivision algorithm recursively subdivides a simple base mesh. The vertices marked are the points subdivided in the figure.

The left-most side is an original mesh as one in the previous section. The second left one is a simple base mesh consisting of four vertices selected from the original. At the first stage, it is subdivided by four simple base meshes as shown in the figure. It seems subdividing the base mesh recursively.

The subdivision may be determined by a specified measure of an error tolerance at each recursion. For example, at second stage, the two base meshes on the upper were subdivided, while the others were not.

3.3 Geometric reduction

Geometric reduction group simplifies an original mesh by removing vertices or polygons [SZL92, AS96] or by merging them [HH93, Ham93, RR96, RO96], if they satisfy a predetermined error measure, which is decided at each reduction stage individually. Figure A.3 shows one example of various algorithms; a vertex removing algorithm.

We assume that the left-most side is an original mesh, where the vertices marked by black circles are to be removed locally. The mesh at the first reduction stage is a simplified mesh. Recursively the simplifiified mesh is to reduced in the same way, as shown at the second and third reduction stage. It should be said that our impoverishing method discussed in *Chapter 3* employed in a similar way.

Another version of geometric reduction algorithms is to simplify an original mesh by employing two offset surfaces; one is the overestimated surface and the other is underestimated [Var94, CVM+96]. The original mesh is regarded as a surface between them. That is, one surface is on the outside of the original

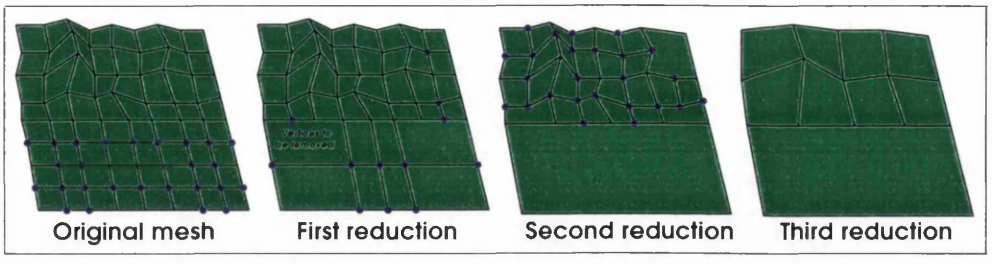


Figure A.4: An example of mesh simplification using geometric reduction

Geometric reduction group simplifies the original mesh by removing vertices or polygons, or merging them. In this Figure, the vertices marked are removed.

and the other is on the inside hierarchically. In this sense, it called an enveloping algorithm. These hierarchical surfaces guarantee the preservation of global topology.

Appendix B

Fundamental Geometry

In this appendix, we introduce the geometric components that contribute to the method for extracting a reconstructed normal in *Chapter 4.*

There are four fundamental geometric components contributing to our new *shape-from-prior-knowledge* method as follows:

- a plane,
- a sphere, and
- two straight lines.

These components can be arranged in $3D$ space as shown in Figure B.1. We will describe them individually.

1 Plane

Provided that a *plane* π is perpendicular to a vector $\vec{L}(l_x,\ l_y,\ l_z)$ and passes through a point $L(l_x,\ l_y,\ l_z)$, we can define the plane π in space as follows:

$$\begin{aligned} \pi \ : \qquad \vec{Lp} \bullet \vec{L} &= 0, \\ (\vec{Op} - \vec{OL}) \bullet \vec{L} &= 0, \\ (x - l_x)l_x + (y - l_y)l_y + (z - l_z)l_z &= 0, \\ xl_x + yl_y + zl_z &= l_x^2 + l_y^2 + l_z^2, \\ xl_x + yl_y + zl_z &= 1, \end{aligned} \tag{B.1}$$

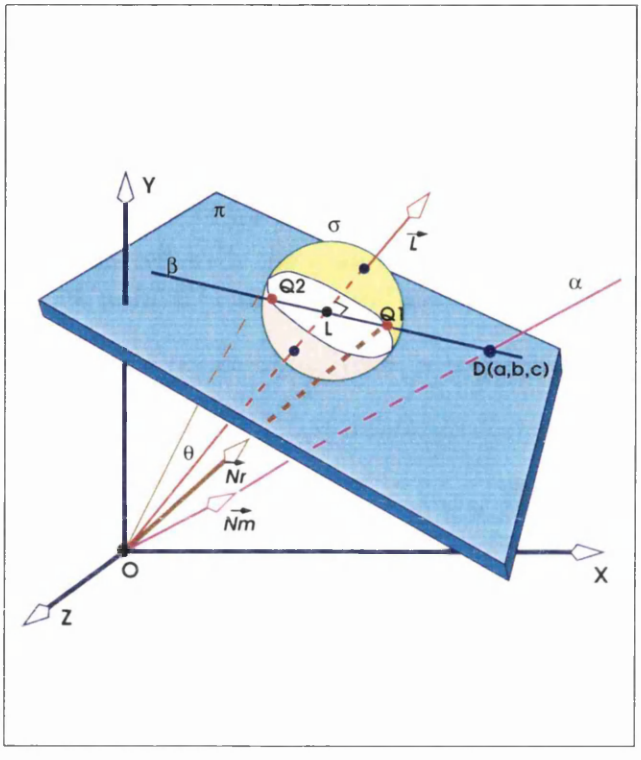


Figure B.1: Fundamental geometric components

The plane π passes through a point $L(l_x, l_y, l_z)$ and it is perpendicular to the vector $\vec{L}(l_x, l_y, l_z)$. The sphere σ has a centre point L and a constant radius R, and is bisected by the plane π. The line α is parallel to $\vec{N_m}$ and passing through the origin O. It meets a point $D(a, b, c)$ with the plane π. The line β passes through two points $D(a, b, c)$ and $L(l_x, l_y, l_z)$ so that it is placed on the plane π.

where $p(x, y, z)$ is an arbitrary point on the plane π, and $\vec{Op}$ and $\vec{OL}$ are position vectors. $\vec{Lp}$ (or $\vec{Op} - \vec{OL}$) is a vector on the plane π, passing through the point L. The distance from the origin O to the plane π is equal to one, provided that $l_x^2 + l_y^2 + l_z^2 = 1$.

2 Sphere

A *sphere* σ can be defined by a radius R and a centre point $L(l_x, l_y, l_z)$ such that an arbitrary point $p(x, y, z)$ on it has the same distance R from the point L. Therefore, the equation of this sphere is written as follows:

$$\sigma : \ |\vec{Op} - \vec{OL}|^2 = (x - l_x)^2 + (y - l_y)^2 + (z - l_z)^2 = R^2, \qquad \text{(B.2)}$$

where $|...|$ represents a magnitude of a vector.

The plane π, perpendicular to the vector $\vec{L}$, bisects the sphere σ. The bisected area forms a 3D circle with a radius of R. It will be a base circle of a space cone with a spread angle θ. The axis of this cone is parallel with $\vec{L}$ and the apex is the origin O.

3 Lines

Now we intend to obtain two equations of straight lines α and β in space respectively. We assume that the line α passes through the origin O and is parallel to a vector $\vec{N_m}(X_m, Y_m, Z_m)$, and that the line β passes through two points $D(a, b, c)$ and $L(l_x, l_y, l_z)$. As shown in Figure B.1, it should be observed that the line α is intersected with the plane π on a point $D(a, b, c)$. The details of computation for the point D can be found in *Appendix C.*

The line α can be written in a vector form as follows:

$$\alpha : \ \vec{Op}(x, y, z) = t\vec{N_m}(X_m, Y_m, Z_m),$$

where t is a real number and $p(x, y, z)$ is an arbitrary point on the line α, and $\vec{Op}$ is a position vector. We can also rewrite it parametrically as follows:

$$\begin{aligned} \alpha : \ x &= tX_m, \\ y &= tY_m, \\ z &= tZ_m. \end{aligned} \tag{B.3}$$

Similarly, the line β can be expressed in the form of a vector as:

$$\begin{aligned} \beta : \ \vec{Op}(x, y, z) &= \vec{OL} + t(\vec{LD}) \\ &= \vec{OL}(l_x, l_y, l_z) + t(\vec{OD}(a, b, c) - \vec{OL}(l_x, l_y, l_z)), \end{aligned}$$

where t is again a real number, $p(x, y, z)$ is an arbitrary point on β, $\vec{Op}$ and $\vec{OL}$ are position vectors. This line β can also be written parametrically as follows:

$$\begin{aligned} \beta : \ x &= l_x + t(a - l_x), \\ y &= l_y + t(b - l_y), \\ z &= l_z + t(c - l_z) \end{aligned} \tag{B.4}$$

Appendix C

Extracting a Reconstructed Normal

In *Chapter 4*, we have briefly derived a method in order to solve our problem, which is to extract a reconstructed normal $\vec{N_r}$ from an intensity value I in a *Lambertian* image and a model normal $\vec{N_m}$. We will give here a full account of the method, which uses the same notations as usual in *Chapter 4*.

1 Point $D(a, b, c)$ Intersecting Plane π and Line α

This section gives a full description of how we obtain a point $D(a,\ b,\ c)$ intersecting the plane π (eq. (B.1)) with the line α (eq. (B.3)) shown in Figure B.1 in *Chapter 4*. If we substitute x, y and z in α for the plane $\pi : \ xl_x + yl_y + zl_z = 1$, we have a new equation for t:

$$\begin{aligned} tX_m l_x + tY_m l_y + tZ_m l_z &= 1, \\ t(X_m l_x + Y_m l_y + Z_m l_z) &= 1, \\ t(\vec{N_m} \bullet \vec{L}) &= 1. \end{aligned}$$

Provided that $X_m l_x + Y_m l_y + Z_m l_z \neq 0$; that is, $\vec{N_m}$ is not perpendicular to

$\vec{L}$, we have t as follows:

$$\begin{aligned} t &= \frac{1}{X_m l_x + Y_m l_y + Z_m l_z} \\ &= \frac{1}{\vec{N_m} \bullet \vec{L}}. \end{aligned}$$

Therefore, by substituting t for the line α (eq. (B.3)), we have the point $D(a,\ b,\ c)$ as follows:

$$\begin{aligned} a &= tX_m = \frac{X_m}{X_m l_x + Y_m l_y + Z_m l_z} = \frac{X_m}{\vec{N_m} \bullet \vec{L}}, \\ b &= tY_m = \frac{Y_m}{X_m l_x + Y_m l_y + Z_m l_z} = \frac{Y_m}{\vec{N_m} \bullet \vec{L}}, \\ c &= tZ_m = \frac{X_m}{X_m l_x + Y_m l_y + Z_m l_z} = \frac{X_m}{\vec{N_m} \bullet \vec{L}}, \end{aligned}$$

where $\vec{N_m} \bullet \vec{L} \neq 0$. We will say that $D(a,\ b,\ c)$ is `decidable`, if D exists; that is, $\vec{N_m} \bullet \vec{L} \neq 0$.

2 Reconstructed Normal $\vec{N_r}$

In solving the shape-from-prior-knowledge problem, we start with three given components at a particular point in an image: an intensity value I, a source vector $\vec{L}$ and a model normal $\vec{N_m}$.

As shown in Figure C.1, a reconstructed normal $\vec{N_r}$ is obtained from a point $D(a,\ b,\ c)$ and a radius R. A model normal $\vec{N_m}$ and a source vector $\vec{L}$ determine the point D according to values of $-1 \leq \vec{N_m} \bullet \vec{L} \leq 1$.

On the other hand, the intensity value I determines the length of radius R which is identical with $\tan(\cos^{-1} I)$ and varies from 0 to ∞ $(0 \leq R \leq \infty)$.

In this section, we will fully describe how a reconstructed normal $\vec{N_r}$ is obtained according to the values of $\vec{N_m} \bullet \vec{L}$ and R.

2.1 Condition 1: $0 < \vec{N_m} \bullet \vec{L} < 1$

In this case, point $D(a,\ b,\ c)$ is `determinable` (exists), therefore a reconstructed normal $\vec{N_r}$ is either $O\vec{Q}_1$ or $O\vec{Q}_2$ as shown in Figure C.1. The points Q_1 and Q_2

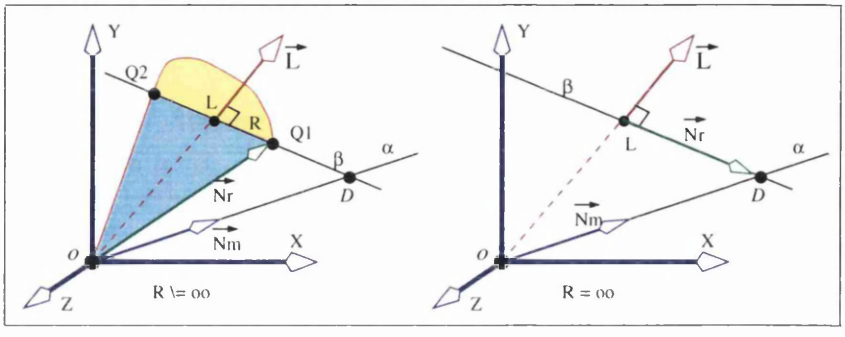


Figure C.1: Extracting $\vec{N}_r$ in case $0 < \vec{N}_m \bullet \vec{L} < 1$

The left picture is an illustration of $R \neq \infty$ ($I \neq 0$). The right one illustrates the case of $R = \infty$ ($I = 0$), in which the reconstructed normal $\vec{N}_r$ is perpendicular to $\vec{L}$.

are points at which sphere σ (eq. (B.2)) and line β (eq. (B.4)) are intersected. Let us substitute x, y and z in the line β (eq. (B.4)) for them in the sphere σ (eq. (B.2)),

$$\begin{aligned}(l_x + t(a - l_x) - l_x)^2 + (l_y + t(a - l_y) - l_y)^2 + (l_z + t(a - l_z) - l_z)^2 &= R^2, \\ t^2((a - l_x)^2 + (b - l_y)^2 + (c - l_z)^2) &= R^2.\end{aligned}$$

Consequently, since $0 < \vec{N}_m \bullet \vec{L} < 1$, we have t as follows:

$$t = \frac{\pm R}{\sqrt{(a - l_x)^2 + (b - l_y)^2 + (c - l_z)^2}},$$

where positive t represents point Q_1 which is the nearer to $\vec{N}_m$, and negative t represents the farther point Q_2. Hence, it is clear that the reconstructed normal $\vec{N}_r$ is $O\vec{Q}_1$ in terms of positive t. From the line β (eq. (B.4)), the point Q_1 can be obtained as follows:

$$\begin{aligned}\vec{N}_r(X_r,\ Y_r,\ Z_r) &= Q_1(x,\ y,\ z) \\ &= (l_x + t(a - l_x),\ l_y + t(b - l_y),\ l_z + t(c - l_z)),\end{aligned} \tag{C.1}$$

where t is positive, that is,

$$t = \frac{R}{\sqrt{(a - l_x)^2 + (b - l_y)^2 + (c - l_z)^2}}. \tag{C.2}$$

It should be stressed that $R \neq \infty$ $(0 < I \leq 1)$ in the above solution $\vec{N_r}$. The left picture in Figure C.1 illustrates this condition.

If $R = \infty$ $(I = 0)$, however the points Q_1 and Q_2 can not be obtained and $\vec{N_r}$ must be perpendicular to $\vec{L}$. Consequently, the reconstructed normal $\vec{N_r}$ is the vector $\vec{LD}$ in this case, as shown on the right picture in Figure C.1. Therefore we have $\vec{N_r}$ as follows:

$$\begin{aligned} \vec{N_r}(X_r,\ Y_r,\ Z_r) &= \vec{LD} \\ &= (a - l_x,\ b - l_y,\ c - l_z). \end{aligned}$$

□ Proof: If $R \to \infty$ and $0 < \vec{N_m} \bullet \vec{L} < 1$, then $t \to \infty$ from the equation (C.2). Let us apply this limit value t to the equation (C.1) as follows:

$$\begin{aligned} \lim_{R\to\infty} \vec{N_r}(X_r,\ Y_r,\ Z_r) &= \lim_{t\to\infty} \vec{N_r}(X_r,\ Y_r,\ Z_r) \\ &= \lim_{t\to\infty} \frac{\vec{N_r}}{t}(X_r,\ Y_r,\ Z_r) \\ &= \lim_{t\to\infty} \vec{N_r}(\frac{X_r}{t},\ \frac{Y_r}{t},\ \frac{Z_r}{t}). \end{aligned}$$

Consequently,

$$\begin{aligned} \lim_{R\to\infty} \vec{N_r} &= \lim_{t\to\infty}(\frac{l_x}{t} + (a - l_x),\ \frac{l_y}{t} + (b - l_y),\ \frac{l_z}{t} + (c - l_z)) \\ &= (a - l_x,\ b - l_y,\ c - l_z), \end{aligned}$$

and hence we have $\vec{LD}$ as $\vec{N_r}$ in the case of $R = \infty$ and $0 < \vec{N_m} \bullet \vec{L} < 1$, as shown on the right picture in Figure C.1.

2.2 Condition 2: $-1 < \vec{N_m} \bullet \vec{L} < 0$

In this case, point $D(a,\ b,\ c)$ is also `determinable` (exists), and points Q_1 and Q_2 can be obtained as mentioned in the previous section. However the reconstructed normal $\vec{N_r}$ is identical with not $\vec{OQ_1}$ but $\vec{OQ_2}$, in terms of negative t. Since the angle between $\vec{N_m}$ and $\vec{L}$ is bigger than $\frac{\pi}{2}$ as shown in Figure C.2, the point closest to the model normal $\vec{N_m}$ is Q_2.

Therefore we have

$$\begin{aligned} \vec{N_r}(X_r,\ Y_r,\ Z_r) &= Q_1(x,\ y,\ z) \\ &= (l_x - t(a - l_x),\ l_y - t(b - l_y),\ l_z - t(c - l_z)), \end{aligned} \tag{C.3}$$

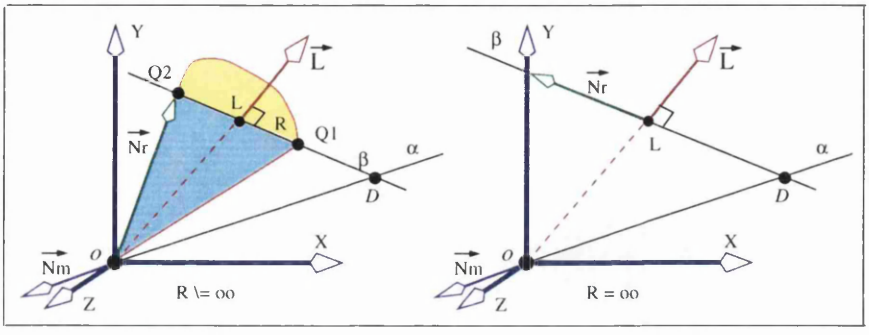


Figure C.2: Extracting $\vec{N_r}$ in case $-1 < \vec{N_m} \bullet \vec{L} < 0$

The left picture is an illustration of $R \neq \infty$ ($I \neq 0$). The right one illustrates the case of $R = \infty$ ($I = 0$), in which the reconstructed normal $\vec{N_r}$ is perpendicular to $\vec{L}$.

where t is identical with the equation (C.2).

If $R = \infty$ ($I = 0$), however the points Q_1 and Q_2 can not be obtained and $\vec{N_r}$ must be perpendicular to $\vec{L}$. Consequently, the reconstructed normal $\vec{N_r}$ is the vector $\vec{DL}$ in this case, as shown on the right picture in Figure C.2. Therefore we have $\vec{N_r}$ as follows:

$$\begin{aligned} \vec{N_r}(X_r,\ Y_r,\ Z_r) &= \vec{DL} \\ &= (l_x - a,\ l_y - b,\ l_z - c). \end{aligned}$$

□ Proof: If $R \to \infty$ and $-1 < \vec{N_m} \bullet \vec{L} < 0$, then $t \to \infty$ from the equation (C.2). Let us apply this limit value t to the equation (C.3) as follows:

$$\begin{aligned} \lim_{R\to\infty} \vec{N_r}(X_r,\ Y_r,\ Z_r) &= \lim_{t\to\infty} \vec{N_r}(X_r,\ Y_r,\ Z_r) \\ &= \lim_{t\to\infty} \vec{N_r}(\frac{X_r}{t},\ \frac{Y_r}{t},\ \frac{Z_r}{t}) \\ &= \lim_{t\to\infty} (\frac{l_x}{t} - (a - l_x),\ \frac{l_y}{t} - (b - l_y),\ \frac{l_z}{t} - (c - l_z)) \\ &= (l_x - a,\ l_y - b,\ l_z - c). \end{aligned}$$

Hence we have $\vec{DL}$ as $\vec{N_r}$ in the case of $R = \infty$ and $-1 < \vec{N_m} \bullet \vec{L} < 0$, as shown on the right picture in Figure C.2.

2.3 Condition 3: $\vec{N_m} \bullet \vec{L} = 0$

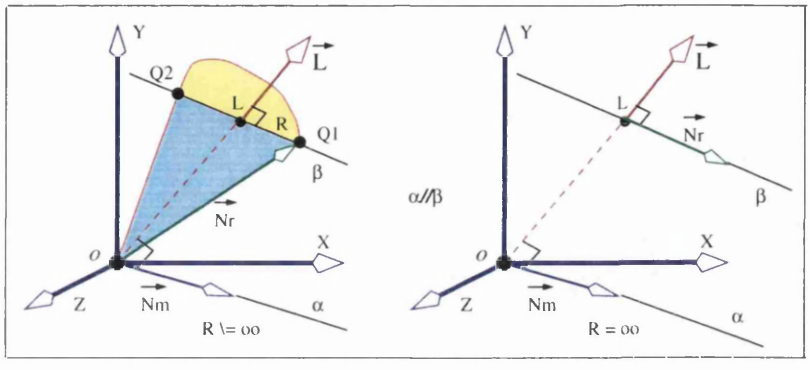


Figure C.3: Extracting $\vec{N_r}$ in case $\vec{N_m} \bullet \vec{L} = 0$

The left picture is an illustration of $R \neq \infty$ ($I \neq 0$). The right one illustrates the case of $R = \infty$ ($I = 0$), in which the reconstructed normal $\vec{N_r}$ is perpendicular to $\vec{L}$.

In this case, since $\vec{N_m}$ is perpendicular to $\vec{L}$, line α parallel with $\vec{N_m}$ never meets with plane π. That is, $D(a,\ b,\ c)$ is `undeterminable`. Consequently we can not determine line β (eq. (B.4)) passing through two points D and L.

However we can redefine an alternative line β which passes through point L and is also parallel to the model normal $\vec{N_m}$ as shown in Figure C.3 as follows:

$$\begin{aligned} \beta: \ x &= l_x + tX_m, \\ y &= l_y + tY_m, \\ z &= l_z + tZ_m. \end{aligned} \tag{C.4}$$

This line β parallel to line α ($\alpha // \beta$) meets two points Q_1 and Q_2, intersecting with the sphere σ (eq. (B.2)), as shown on the left picture in Figure C.3. To determine t, provided that we replace x, y and z in the sphere σ (eq. (B.2)) by the equation (C.4), we can rewrite it as follows:

$$\begin{aligned} (l_x + tX_m - l_x)^2 + (l_y + tY_m - l_y)^2 + (l_z + tZ_m - l_z)^2 &= R^2, \\ t^2(X_m^2 + Y_m^2 + Z_m^2) &= R^2. \end{aligned}$$

Therefore we have

$$t = \frac{\pm R}{\sqrt{X_m^2 + Y_m^2 + Z_m^2}}, \tag{C.5}$$

where $R \neq \infty$ $(0 < I \leq 1)$. Positive t represents the point Q_1 which is the closer to $\vec{N_m}$ and negative t gives the farther point Q_2. It is clear that the reconstructed normal $\vec{N_r}$ is identical with $O\vec{Q}_1$.

Therefore we can obtain the reconstructed normal $\vec{N_r}$ from equations (C.4) and (C.5) as follows:

$$\begin{aligned} \vec{N_r}(X_r,\ Y_r,\ Z_r) &= O\vec{Q}_1(x,\ y,\ z) \\ &= (l_x + tX_m,\ l_y + tY_m,\ l_z + tZ_m), \end{aligned} \tag{C.6}$$

where t is positive and $R \neq \infty$ $(0 < I \leq 1)$. It should be stressed that negative t is not used, because we always select the point Q_1 in the direction of the model normal $\vec{N_m}$.

On the other hand, if $R = \infty$ $(I = 0)$, points Q_1 and Q_2 cannot be obtained and $\vec{N_r}$ must be perpendicular to $\vec{L}$. Consequently, the reconstructed normal $\vec{N_r}$ is the vector $\vec{N_m}$ here, as shown on the right picture in Figure C.3. That is, we have $\vec{N_r}$ as follows:

$$\vec{N_r}(X_r,\ Y_r,\ Z_r) = \vec{N_m}(X_m,\ Y_m,\ Z_m).$$

□ Proof: If $R \to \infty$ and $\vec{N_m} \bullet \vec{L} = 0$, then $t \to \infty$ from the equation (C.5) with positive t. Let us apply this limit value t to the equation (C.6) as follows:

$$\begin{aligned} \lim_{R\to\infty} \vec{N_r}(X_r,\ Y_r,\ Z_r) &= \lim_{t\to\infty} \vec{N_r}(X_r,\ Y_r,\ Z_r) \\ &= \lim_{t\to\infty} \vec{N_r}(\frac{X_r}{t},\ \frac{Y_r}{t},\ \frac{Z_r}{t}) \\ &= \lim_{t\to\infty} (\frac{l_x}{t} + X_m,\ \frac{l_y}{t} + Y_m,\ \frac{l_z}{t} + Z_m) \\ &= (X_m,\ Y_m,\ Z_m). \end{aligned}$$

Hence we have $\vec{N_m}$ as $\vec{N_r}$ in the case of $R = \infty$ and $\vec{N_m} \bullet \vec{L} = 0$, as shown on the right picture in Figure C.3.

2.4 Condition 4: $\vec{N_m} \bullet \vec{L} = \pm 1$

This condition occurs when a model normal $\vec{N_m}$ is parallel to a source vector $\vec{L}$. In this case, point $D(a, b, c)$ intersecting line α with plane π is identical to point $L(l_x, l_y, l_z)$ as shown on the right picture in Figure C.4.

So we cannot decide line β (eq. (B.4)) passing through two points L and D. However if we can take an alternative model normal $\vec{N_m'}$, which is not parallel,

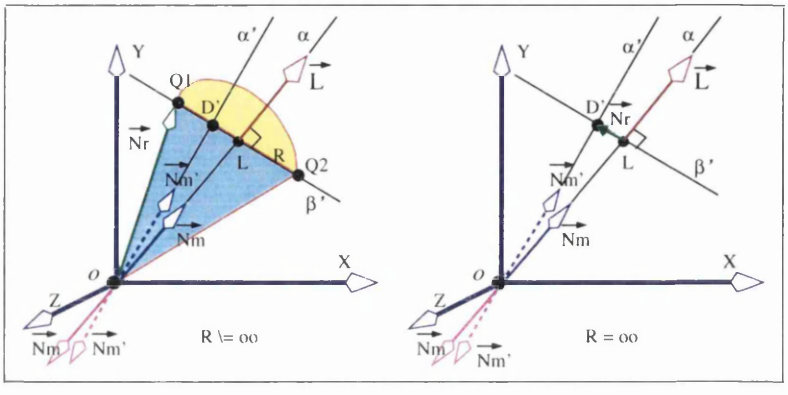


Figure C.4: Extracting $\vec{N_r}$ in case $\vec{N_m} \bullet \vec{L} = \pm 1$

The left picture is an illustration of $R \neq \infty$ ($I \neq 0$). The right picture illustrates the case of $R = \infty$ ($I = 0$), in which the reconstructed normal $\vec{N_r}$ is perpendicular to $\vec{L}$.

but approximate to $\vec{L}$, the line α' parallel to the model normal $\vec{N_m'}$ gives the point $D'(a',\ b',\ c')$ on plane π as shown on the right picture in Figure C.4.

Now we can decide line β' passing through two points L and D'. Therefore this condition becomes one of the above cases mentioned in the previous sections.

3 Summaries of Extracting $\vec{N_r}$

Table C.1 shows summaries how the reconstructed normal $\vec{N_r}(X_r,\ Y_r,\ Z_r)$ can be calculated, given an intensity value $0 \leq I \leq 1$, a model normal $\vec{N_m}(X_m,\ Y_m,\ Z_m)$, and a source vector $\vec{L}(l_x,\ l_y,\ l_z)$ at a particular point in an image. The values of t_1, t_2, R and $D(a,\ b,\ c)$ in this table are as follows:

$$
\begin{aligned}
t_1 &= \frac{R}{\sqrt{(a-l_x)^2 + (b-l_y)^2 + (c-l_z)^2}}\ , \\
t_2 &= \frac{R}{\sqrt{X_m^2 + Y_m^2 + Z_m^2}}\ , \\
D &= \left(\frac{X_m}{\vec{N_m} \bullet \vec{L}},\ \frac{X_m}{\vec{N_m} \bullet \vec{L}},\ \frac{X_m}{\vec{N_m} \bullet \vec{L}}\right)\ , \\
R &= \tan(\cos^{-1} I)\ .
\end{aligned}
$$

$\vec{N_m} \bullet \vec{L}$	$R \neq \infty$ $(0 < I \leq 1)$	$R = \infty$ $(I = 0)$
$0 < \vec{N_m} \bullet \vec{L} < 1$	$\vec{N_r} = \begin{pmatrix} l_x + t_1(a - l_x) \\ l_y + t_1(b - l_y) \\ l_z + t_1(c - l_z) \end{pmatrix}^{-1}$	$\vec{N_r} = \begin{pmatrix} a - l_x \\ b - l_y \\ c - l_z \end{pmatrix}^{-1}$
$-1 < \vec{N_m} \bullet \vec{L} < 0$	$\vec{N_r} = \begin{pmatrix} l_x - t_1(a - l_x) \\ l_y - t_1(b - l_y) \\ l_z - t_1(c - l_z) \end{pmatrix}^{-1}$	$\vec{N_r} = \begin{pmatrix} l_x - a \\ l_y - b \\ l_z - c \end{pmatrix}^{-1}$
$\vec{N_m} \bullet \vec{L} = 0$	$\vec{N_r} = \begin{pmatrix} l_x + t_2 X_m \\ l_y + t_2 Y_m \\ l_z + t_2 Z_m \end{pmatrix}^{-1}$	$\vec{N_r} = \begin{pmatrix} X_m \\ Y_m \\ Z_m \end{pmatrix}^{-1}$
$\vec{N_m} \bullet \vec{L} = \pm 1$	Select $\vec{N_m'}$ to go back one of the above conditions	

Table C.1: Summaries of $\vec{N_r}$

Appendix D

Quaternion Space

Quaternions provide a fundamental means to describe the rotation of a vector in $3D$ space [Sho85, Sho87, Ple89, BCGH92] so that they are frequently used to solve rotation problems in computer graphics and animation. Especially quaternion multiplication contributes to solving the rotation problems.

In *Chapter 5*, we newly applied quaternion multiplication in order to rotate a surface patch (triangle). These rotations allow us to reconstruct an approximation of the original surface from a single image. So we will here give a full account of quaternion multiplication.

1 Properties of Quaternions

There are several ways to define a quaternion such as a vector or a complex number. Amongst these, we adopt a way that represents it as an element consisting of a scalar and a vector part for our purposes. The former may represent an amount of angle to rotated, a so-called `rotation angle`, the latter may represent a `rotation axis`.

The rotation of a vector can be described by a rotation angle and a rotation axis in space. As shown in Figure 5.3, the angle ϕ can be considered as a rotation angle and the vector $\vec{n}$ as a rotation axis.

property	definition
representation	$q = q(\texttt{scalar, vector}) = q(\texttt{angle, axis}) = q(s, \vec{v})$
inverse (conjugate)	$q^{-1} = q^{-1}(s, -\vec{v})$
magnitude	$\|q\|^2 = qq^{-1} = s^2 + \|\vec{v}\|^2$
multiplication	$q_1q_2 = q_1(s_1, \vec{v_1})q_2(s_2, \vec{v_2}) = q_1q_2(s_1s_2 - \vec{v_1} \bullet \vec{v_2}, s_1\vec{v_2} + s_2\vec{v_1} + \vec{v_1} \times \vec{v_2})$
associative	$q_1q_2q_3 = (q_1q_2)q_3 = q_1(q_2q_3)$
NO commutative	$q_1q_2 \neq q_2q_1$
unit quaternion	q; one satisfies that $\|q\|^2 = 1$
pure quaternion	$p = p(0, \vec{v})$; one that has no scalar part.
rotation	$p'(0, \vec{v'}) = qpq^{-1} = q(\cos\frac{\phi}{2}, \vec{n}\sin\frac{\phi}{2})p(0, \vec{v})q^{-1}(\cos\frac{\phi}{2}, -\vec{n}\sin\frac{\phi}{2}) = qpq^{-1}(0, \vec{v}\cos\phi + (1-\cos\phi)\vec{n}(\vec{n} \bullet \vec{v}) + \sin\phi(\vec{n} \times \vec{v}))$

Table D.1: Basic properties of quaternions

By this reason, we define a quaternion q for our purposes as follows:

$$\begin{aligned} q &= q(\texttt{scalar, vector}), \\ &= q(\texttt{angle, axis}), \\ &= q(s, \vec{v}), \\ &= q(\cos\frac{\phi}{2}, \vec{n}\sin\frac{\phi}{2}), \end{aligned}$$

where $|q|^2 = s^2 + |\vec{v}|^2 = 1$, provided that $|\vec{n}|^2 = 1$. This kind of quaternion is called a `unit quaternion`. If no scalar part, it is called a `pure quaternion` such as $p(0, \vec{v})$.

The multiplication of two quaternions $q_1(s_1, \vec{v_1})$ and $q_2(s_2, \vec{v_2})$ is defined by:

$$\begin{aligned} q_1q_2 &= q_1(s_1, \vec{v_1})q_2(s_2, \vec{v_2}) \\ &= q_1q_2(s_1s_2 - \vec{v_1} \bullet \vec{v_2}, s_1\vec{v_2} + s_2\vec{v_1} + \vec{v_1} \times \vec{v_2}), \end{aligned}$$

where $\bullet$ and $\times$ represent a dot and cross product of two vectors respectively. The other properties of quaternion are summarised in Table D.1.

2 Rotation using Quaternions

Provided that a rotation angle is given by ϕ and a rotation axis by $\vec{n}$, a rotation of a vector $\vec{v}$ can be defined by the multiplication in terms of three quaternions as shown in Table D.1: a unit quaternion $q(\cos\frac{\phi}{2},\ \vec{n}\sin\frac{\phi}{2})$, an inverse of the unit quaternion $q^{-1}(\cos\frac{\phi}{2},\ -\vec{n}\sin\frac{\phi}{2})$, and a pure quaternion $p(0,\ \vec{v})$ as follows:

$$\begin{aligned} qpq^{-1} &= (\cos\frac{\phi}{2},\ \vec{n}\sin\frac{\phi}{2})(0,\ \vec{v})(\cos\frac{\phi}{2},\ -\vec{n}\sin\frac{\phi}{2}) \\ &= (0,\ \vec{v}\cos\phi + (1-\cos\phi)\vec{n}(\vec{n}\bullet\vec{v}) + \sin\phi(\vec{n}\times\vec{v})). \end{aligned}$$

The pure quaternion $p(0,\ \vec{v})$ is obtained from the vector $\vec{v}$ to be rotated. The q and q^{-1} are also obtained from the rotation angle ϕ and the rotation axis $\vec{n}$. The vector part of qpq^{-1} is a rotated vector $\vec{v'}$.

Again, we can consequently formulate a rotated vector $\vec{v'}$ in terms of the original vector $\vec{v}$, the angle ϕ, and the axis $\vec{n}$ fully:

$$\vec{v'} = \vec{v}\cos\phi + (1-\cos\phi)\vec{n}(\vec{n}\bullet\vec{v}) + \sin\phi(\vec{n}\times\vec{v})$$

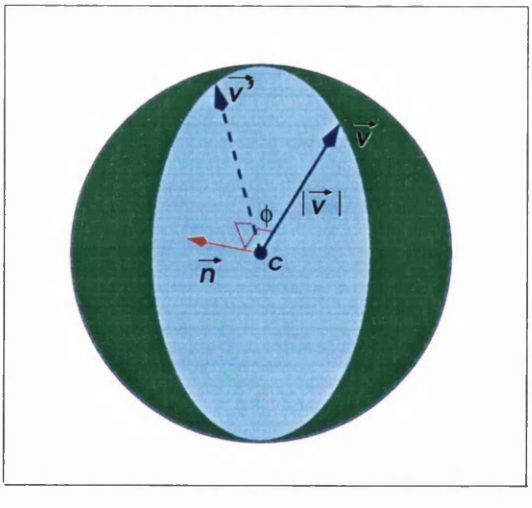


Figure D.1: Rotation sphere

Figure D.1 shows a rotation example of a vector $\vec{v}$ about an angle ϕ and an axis $\vec{n}$. The vector $\vec{v'}$ is a rotated vector. If we employ an arbitray angle and axis to rotate the vector $\vec{v}$, rotated vectors form a sphere with a centre c and a radius $|\vec{v}|$.

□ Proof: We are going to verify qpq^{-1}. Let the scalar part of a quaternion Q be $S(Q)$ and the vector part be $V(Q)$. Multiplication of quaternions is associative but not commutative. So we can compute qpq^{-1} through the associative rule:

$qpq^{-1} = (qp)q^{-1}$. By the definition of multiplication, qp can be obtained as follows:

$$\begin{aligned} qp &= (\cos\frac{\phi}{2},\ \vec{n}\sin\frac{\phi}{2})(0,\ \vec{v}) \\ &= (-\sin\frac{\phi}{2}\vec{n}\bullet\vec{v},\ \ \vec{v}\cos\frac{\phi}{2}+\sin\frac{\phi}{2}(\vec{n}\times\vec{v})). \end{aligned}$$

Therefore qpq^{-1} may be given as:

$$qpq^{-1} = (-\sin\frac{\phi}{2}\vec{n}\bullet\vec{v},\ \vec{v}\cos\frac{\phi}{2}+\sin\frac{\phi}{2}(\vec{n}\times\vec{v}))(\cos\frac{\phi}{2},\ \ -\vec{n}\sin\frac{\phi}{2}).$$

Now let us calculate the scalar and vector part of qpq^{-1} separately. The scalar part $S(qpq^{-1})$ can be obtained by:

$$\begin{aligned} S(qpq^{-1}) &= -\sin\frac{\phi}{2}\cos\frac{\phi}{2}(\vec{n}\bullet\vec{v})+(\vec{v}\cos\frac{\phi}{2}+\sin\frac{\phi}{2}(\vec{n}\times\vec{v}))\bullet(\sin\frac{\phi}{2}\vec{n}) \\ &= \sin^2\frac{\phi}{2}\vec{n}\bullet(\vec{n}\times\vec{v}) \\ &= 0, \end{aligned}$$

where $\vec{n}\bullet(\vec{n}\times\vec{v}) = 0$, since $\vec{n}$ is perpendicular to $\vec{n}\times\vec{v}$.

Before computing the vector part $V(qpq^{-1})$, it is necessary to remind ourselves of several rules of vectors:

$$\begin{aligned} (\vec{A}+\vec{B})\times\vec{C} &= \vec{A}\times\vec{C}+\vec{B}\times\vec{C}, \\ \vec{A}\times\vec{B} &= -\vec{B}\times\vec{A}, \\ (\vec{A}\times\vec{B})\times\vec{C} &= (\vec{A}\bullet\vec{C})\times\vec{B}-(\vec{B}\bullet\vec{C})\times\vec{A}, \end{aligned}$$

Using those rules of vectors, the vector part $V(qpq^{-1})$ can be obtained as

follows:

$$\begin{aligned}
V(qpq^{-1}) &= (-\sin\tfrac{\phi}{2}\vec{n}\bullet\vec{v})(-\vec{n}\sin\tfrac{\phi}{2}) + (\cos\tfrac{\phi}{2})(\vec{v}\cos\tfrac{\phi}{2} + \sin\tfrac{\phi}{2}(\vec{n}\times\vec{v})) + \\
&\quad (\vec{v}\cos\tfrac{\phi}{2} + \sin\tfrac{\phi}{2}(\vec{n}\times\vec{v}))\times(-\vec{n}\sin\tfrac{\phi}{2}) \\
&= \sin^2\tfrac{\phi}{2}(\vec{n}\bullet\vec{v})\vec{n} + \cos^2\tfrac{\phi}{2}\vec{v} + \cos\tfrac{\phi}{2}\sin\tfrac{\phi}{2}(\vec{n}\times\vec{v}) - \\
&\quad \cos\tfrac{\phi}{2}\sin\tfrac{\phi}{2}(\vec{v}\times\vec{n}) - \sin^2\tfrac{\phi}{2}(\vec{n}\times\vec{v})\times\vec{n} \\
&= \sin^2\tfrac{\phi}{2}(\vec{n}\bullet\vec{v})\vec{n} + \cos^2\tfrac{\phi}{2}\vec{v} + 2\cos\tfrac{\phi}{2}\sin\tfrac{\phi}{2}(\vec{n}\times\vec{v}) - \\
&\quad \sin^2\tfrac{\phi}{2}((\vec{n}\bullet\vec{n})\vec{v} - (\vec{v}\bullet\vec{n})\vec{n}) \\
&= (\cos^2\tfrac{\phi}{2} - \sin^2\tfrac{\phi}{2}(\vec{n}\bullet\vec{n}))\vec{v} + 2\sin^2\tfrac{\phi}{2}(\vec{n}\bullet\vec{v})\vec{n} + \\
&\quad 2\cos\tfrac{\phi}{2}\sin\tfrac{\phi}{2}(\vec{n}\times\vec{v}) \\
&= \vec{v}\cos\phi + (1-\cos\phi)(\vec{n}\bullet\vec{v})\vec{n} + \sin\phi(\vec{n}\times\vec{v}).
\end{aligned}$$

Consequently, qpq^{-1} turns to be a pure quaternion as follows:

$$qpq^{-1} = (0,\ V(qpq^{-1})).$$

Appendix E

Actual Values of Error Estimators for Reconstructed Faces

In this appendix, we can find actual values of *AVID*s and *SDID*s introduced in *Chapter 6*. These values are represented in the form of a table. There are five tables for each experiment conducted in that chapter. The first table represents the values for variously impoverished faces from *me20* to *me90Morp* in 40 iterations.

The other tables contain experimental values for each reconstruction album. These tables are accompanied by charts. The values in a table were obtained in only 20 iterations, because the number is sufficient to show the convergent trend of our reconstruction method.

The experimental values are in a sense more precise than the accuracy associated with any single intensity value in an image. For this reason, we report them using two digits of decimal accuracy.

1 Table for variously impoverished faces

Values shown in Table E.1 are the actual *AVID* and *SDID* values between two images, *Oimage* and *Rimage*. They were observed in the reconstruction process for variously impoverished faces from *me20* to *me90Morp*. These impoverished faces together with reconstructed faces are shown in Figures 5.1 and 6.2.

Highlighted entries in the table are maximum and minimum values of the

	Oimage - Rimage[k]									
	me20		me40		me60		me80		me90Morp	
K	Average	Deviation	Average	Deviation	Average	Deviation	Average	Deviation	Average	Deviation
0	**7.49**	**19.34**	**8.65**	**20.30**	**10.17**	**21.54**	**14.50**	**27.37**	**20.91**	**36.99**
1	6.28	18.34	7.11	18.89	8.30	19.68	12.35	24.65	18.23	33.50
2	5.70	17.93	6.35	18.30	7.30	18.84	10.99	23.10	16.47	31.49
3	5.39	17.72	5.93	18.00	6.73	18.39	10.06	22.08	15.24	30.15
4	5.22	17.61	5.68	17.83	6.37	18.14	9.40	21.35	14.36	29.21
5	5.12	17.54	5.53	17.74	6.14	17.99	8.91	20.83	13.67	28.52
6	5.07	17.51	5.44	17.69	5.99	17.90	8.55	20.44	13.12	27.98
7	**5.05**	17.50	5.39	**17.66**	5.91	17.85	8.26	20.13	12.69	27.59
8	5.06	**17.49**	**5.37**	**17.66**	**5.86**	17.83	8.05	19.91	12.35	27.30
9	5.09	17.51	5.38	17.67	**5.86**	**17.82**	7.89	19.75	12.08	27.08
10	5.15	17.53	5.42	17.69	5.87	**17.82**	7.76	19.63	11.85	26.90
11	5.21	17.56	5.48	17.73	5.90	17.84	7.67	19.55	11.66	26.75
12	5.29	17.59	5.55	17.77	5.94	17.86	7.58	19.47	11.51	26.63
13	5.38	17.64	5.62	17.82	5.99	17.89	7.51	19.41	11.37	26.51
14	5.48	17.69	5.69	17.86	6.05	17.91	7.47	19.35	11.26	26.41
15	5.56	17.73	5.77	17.90	6.12	17.96	7.44	19.31	11.16	26.30
16	5.66	17.78	5.85	17.95	6.19	17.99	**7.42**	**19.28**	11.09	26.23
17	5.75	17.83	5.95	18.00	6.24	18.02	7.43	**19.28**	11.02	26.15
18	5.86	17.90	6.05	18.08	6.32	18.08	7.45	19.30	10.94	26.07
19	5.95	17.95	6.14	18.12	6.39	18.12	7.45	19.31	10.89	26.00
20	6.03	18.00	6.22	18.18	6.45	18.16	7.44	19.30	10.85	25.95
21	6.12	18.06	6.32	18.25	6.53	18.22	7.44	19.30	10.81	25.92
22	6.23	18.13	6.41	18.31	6.60	18.28	7.47	19.32	10.78	25.90
23	6.30	18.16	6.50	18.37	6.66	18.31	7.50	19.31	10.76	25.90
24	6.38	18.21	6.57	18.43	6.73	18.37	7.53	19.32	10.73	25.89
25	6.48	18.30	6.64	18.47	6.79	18.42	7.54	19.34	10.70	25.88
26	6.57	18.36	6.71	18.51	6.86	18.44	7.59	19.39	10.67	25.88
27	6.64	18.39	6.80	18.59	6.95	18.50	7.61	19.40	10.67	25.89
28	6.73	18.45	6.87	18.64	7.01	18.53	7.60	19.39	10.66	25.86
29	6.80	18.51	6.93	18.66	7.05	18.56	7.61	19.40	10.65	25.87
30	6.85	18.54	6.98	18.68	7.09	18.60	7.64	19.44	10.67	25.90
31	6.90	18.57	7.03	18.71	7.12	18.64	7.65	19.43	10.65	25.88
32	6.95	18.59	7.08	18.73	7.15	18.66	7.66	19.44	10.60	25.84
33	6.98	18.61	7.13	18.76	7.17	18.67	7.70	19.48	10.61	25.84
34	7.00	18.63	7.15	18.77	7.20	18.68	7.70	19.50	10.62	25.84
35	7.04	18.65	7.19	18.79	7.22	18.70	7.69	19.48	10.60	25.82
36	7.06	18.65	7.24	18.83	7.27	18.71	7.71	19.50	10.56	25.80
37	7.11	18.69	7.27	18.84	7.31	18.74	7.72	19.49	10.54	25.81
38	7.15	18.71	7.28	18.87	7.35	18.77	7.73	19.50	10.52	25.80
39	7.16	18.72	7.32	18.91	7.41	18.82	7.74	19.50	10.51	**25.78**
40	7.21	18.74	7.36	18.94	7.45	18.85	7.75	19.51	**10.50**	**25.78**

Table E.1: Experimental values for variously impoverished faces

*AVID*s and *SDID*s. The maximum values are all observed at the first rows. They correspond to the impoverished faces given initially. The minimum values are shifted down according to the impoverished faces. The minimum values for *me90Morp* are observed at the 40^{th} iteration, while those of the others are obtained within 20 iterations.

In the table, the left most columns (K's) represent the number of iterations. *Oimage* stands for an input face image, and *Rimage[k]* is an image of a reconstructed face at the k^{th} iteration. Charts obtained from this table are depicted in Figures 6.4 and 6.5 in *Chapter 6.*

2 Album1 : from four different objects

2.1 Table of averages and standard deviations of intensity differences

Table E.2 represents the experimental values for the statistical estimators observed in the reconstruction process for *john80*, *stephane80*, *hand80*, and *instep80*. The impoverished and reconstructed objects are shown in Figure 6.6. Highlighted numbers represent the maximum and minimum values of *AVID*s and *SDID*s in the process.

The maximum values are observed at the first rows. They were obtained from the impoverished models before being processed. The minimum values are shown around the 20^{th} iteration. The values decrease as the iterations progress. This indicates that reconstructed objects converge on the original faces represented by the input images in the process.

2.2 Averages of intensity errors: AVID

The curves in Figure E.1 were plotted by the *AVID* values in Table E.2. They may be characteristic of *AVID*s of 80% impoverished faces. They rapidly drop at the beginning, and then slowly decrease as the process continues. The curve for *stephane80* is slightly more stretched than the others.

From the charts, average intensity errors continuously decrease. Therefore, our reconstruction method improves the impoverished faces in *Album1* to approach the original faces in each input image.

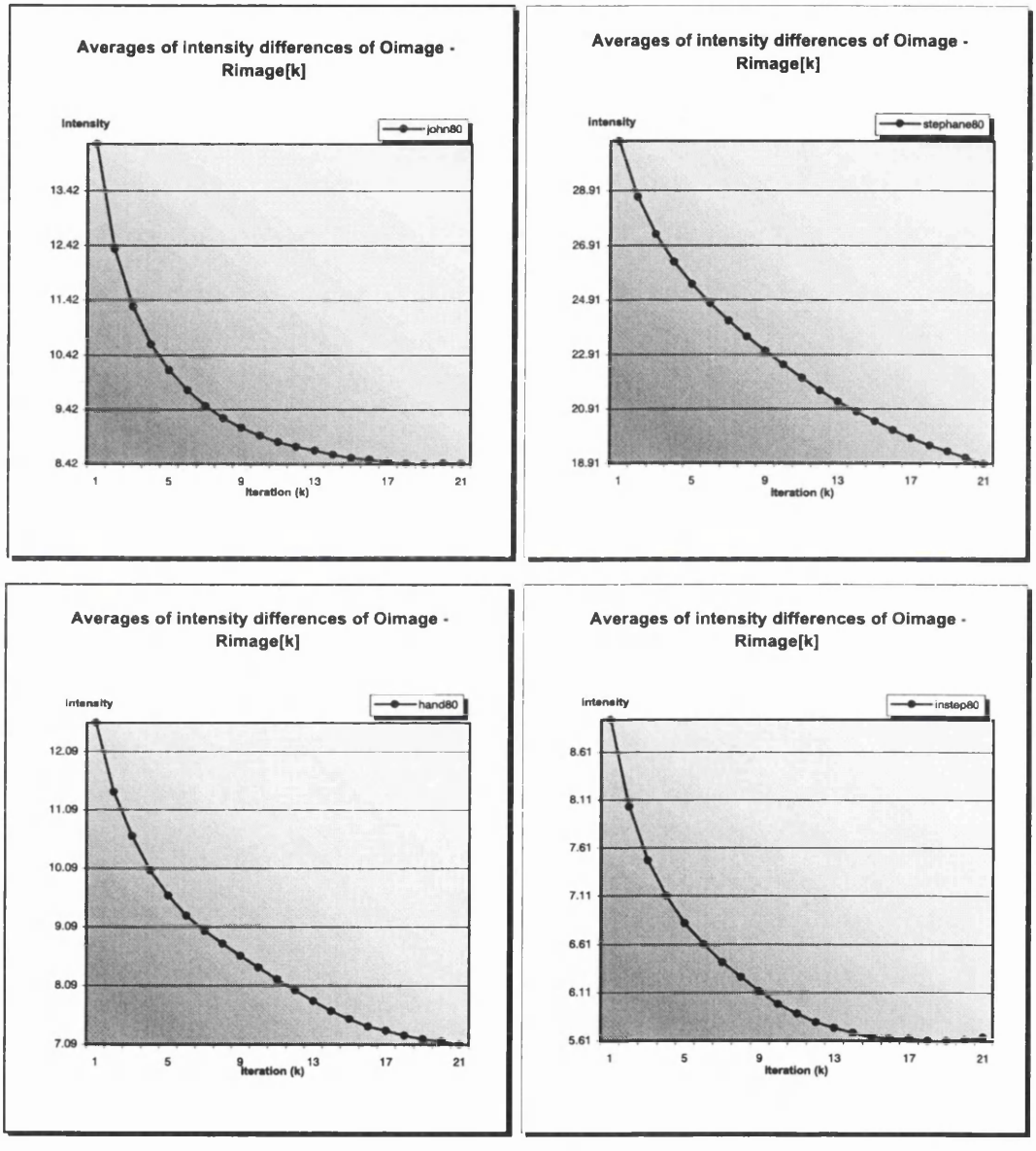


Figure E.1: *AVID* charts for *Album1*

2.3 Standard deviations of intensity erros: SDID

The curves in Figure E.2 were plotted by the *SDID* values in Table E.2. They are similar to the curves with respect to the *AVID* values shown in Figure E.1. As the iteration goes on, they drop rapidly at the beginning and then slowly decrease. That is, standard deviations from each *AVID* value decrease. Consequently, it is clear that our reconstruction method is stable for *Album1*.

3 Album2 : from a half cylinder and ovoid

3.1 Table of averages and standard deviations of intensity differences

Table E.3 shows the experimental values of *AVID*s and *SDID*s. They were observed in the reconstruction process for a half-cylindrical face[1], as shown in Figure 6.8. Highlighted numbers are minima and maxima.

The first rows in the table are the maximum values. They correspond to the impoverished faces before being processed. The minimum values are observed at the 20^{th} iteration. The values between minima and maxima decrease as the iterations go on. This implies that the reconstructed faces converge.

It is noticable that the minimum values in *Album2* are larger than the maxima of *Album1*. For example, the minimum *AVID* of 47.83 for *johnCL* is larger than the maximum *AVID* of 14.27 for *john80*. This difference is due to a lack of prior knowledge. The reconstruction process for *Album1* started with at least 20% prior knowledge, but *Album2* was processed with less knowledge at the beginning.

3.2 Averages of intensity errors: AVID

The curves in Figure E.3 were generated by the *AVID* values in Table E.3. On the whole, they decrease linearly in 20 iterations. This linear decrease is characteristic of the reconstruction starting from an impoverished face that has almost no prior knowledge about the original face. We can find the *AVID* curves for a half ovoid in Figure E.4.

[1]Similarly, the experimental values for a half-ovoid are shown in Table E.4. In addition, the curves corresponding to those values are given in Figures E.4 and E.6 as well.

	Oimage - Rimage[k]							
	john80		stephane80		hand80		instep80	
K	Average	Deviation	Average	Deviation	Average	Deviation	Average	Deviation
0	14.27	26.62	30.72	38.74	12.59	31.33	8.95	24.44
1	12.37	23.03	28.69	35.83	11.40	29.18	8.04	22.12
2	11.31	21.35	27.33	34.06	10.64	27.86	7.48	20.85
3	10.62	20.30	26.31	32.78	10.06	26.69	7.11	20.02
4	10.14	19.59	25.50	31.76	9.62	25.82	6.83	19.42
5	9.77	19.06	24.79	30.89	9.28	25.19	6.61	18.93
6	9.49	18.66	24.15	30.14	9.02	24.73	6.42	18.53
7	9.27	18.35	23.58	29.45	8.80	24.37	6.27	18.23
8	9.09	18.10	23.05	28.83	8.60	24.04	6.12	17.92
9	8.95	17.90	22.54	28.26	8.40	23.69	5.99	17.70
10	8.83	17.73	22.05	27.74	8.20	23.36	5.89	17.53
11	8.74	17.61	21.60	27.27	8.01	23.04	5.80	17.38
12	8.67	17.51	21.19	26.86	7.83	22.77	5.74	17.26
13	8.60	17.41	20.81	26.50	7.66	22.54	5.69	17.18
14	8.54	17.32	20.46	26.15	7.52	22.35	5.65	17.12
15	8.50	17.24	20.14	25.85	7.40	22.21	5.63	17.02
16	8.46	17.18	19.84	25.58	7.32	22.12	5.63	17.03
17	8.43	17.13	19.58	25.33	7.24	22.01	5.61	16.98
18	8.42	17.12	19.35	25.11	7.18	21.92	5.61	16.97
19	8.44	17.12	19.12	24.92	7.14	21.88	5.62	17.01
20	8.43	17.11	18.91	24.75	7.09	21.82	5.65	17.07

Table E.2: Experimental values for *Album1*

	Oimage - Rimage[k]					
	meCL		johnCL		stephaneCL	
K	Average	Deviation	Average	Deviation	Average	Deviation
0	49.44	68.49	61.46	74.83	38.29	50.98
1	47.75	67.54	60.05	73.93	36.96	48.89
2	46.62	66.89	59.12	73.30	35.92	47.32
3	45.74	66.34	58.31	72.73	35.06	46.09
4	44.96	65.83	57.56	72.18	34.31	45.05
5	44.22	65.28	56.85	71.64	33.61	44.12
6	43.48	64.69	56.15	71.10	32.96	43.30
7	42.72	64.10	55.43	70.55	32.34	42.56
8	42.01	63.58	54.70	70.00	31.74	41.89
9	41.35	63.09	53.96	69.44	31.16	41.27
10	40.73	62.61	53.19	68.81	30.59	40.70
11	40.18	62.19	52.45	68.21	30.03	40.15
12	39.71	61.81	51.76	67.68	29.49	39.62
13	39.29	61.45	51.11	67.18	28.98	39.14
14	38.90	61.10	50.52	66.72	28.50	38.68
15	38.55	60.77	49.97	66.27	28.05	38.28
16	38.20	60.41	49.46	65.84	27.65	37.91
17	37.87	60.05	49.01	65.45	27.29	37.58
18	37.58	59.70	48.59	65.10	26.97	37.30
19	37.30	59.37	48.20	64.78	26.70	37.06
20	37.04	59.07	47.83	64.46	26.47	36.85

Table E.3: Experimental values for *Album2*

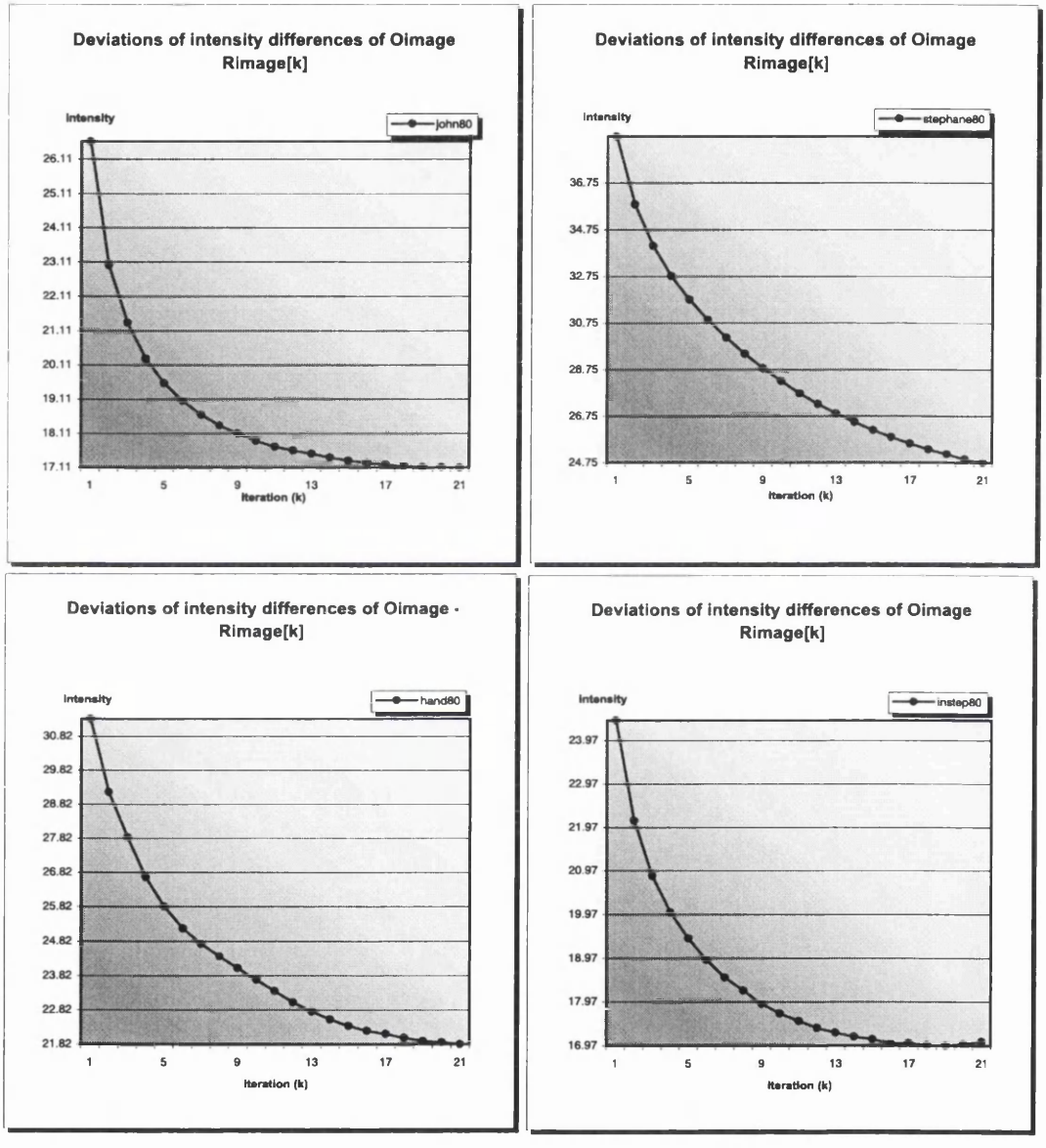


Figure E.2: *SDID* charts for *Album1*

	Oimage - Rimage[k]					
	meOV		johnOV		stephaneOV	
K	Average	Deviation	Average	Deviation	Average	Deviation
0	**24.43**	**43.43**	**30.00**	**47.41**	**34.59**	**54.02**
1	23.60	42.77	29.21	46.42	34.18	53.63
2	23.00	42.37	28.56	45.71	33.80	53.28
3	22.50	42.06	28.02	45.18	33.44	52.99
4	22.10	41.84	27.55	44.78	33.06	52.71
5	21.71	41.61	27.09	44.39	32.68	52.48
6	21.40	41.48	26.68	44.11	32.31	52.27
7	21.12	41.33	26.28	43.83	31.97	52.10
8	20.88	41.20	25.94	43.64	31.60	51.89
9	20.67	41.10	25.63	43.45	31.31	51.78
10	20.52	41.03	25.31	43.24	31.00	51.61
11	20.40	40.99	25.06	43.19	30.74	51.50
12	20.25	40.83	24.84	43.10	30.56	51.48
13	20.19	40.80	24.68	43.09	30.38	51.43
14	20.11	40.70	24.50	43.03	30.23	51.40
15	20.10	40.65	24.36	43.01	30.12	51.41
16	20.08	40.60	24.28	**43.00**	30.00	51.37
17	**20.06**	**40.52**	24.20	43.03	29.92	**51.35**
18	20.11	40.56	24.16	43.03	29.89	51.39
19	20.15	40.53	24.08	43.01	**29.86**	51.37
20	20.25	40.65	**24.06**	43.06	29.89	51.40

Table E.4: Experimental values for a half ovoid

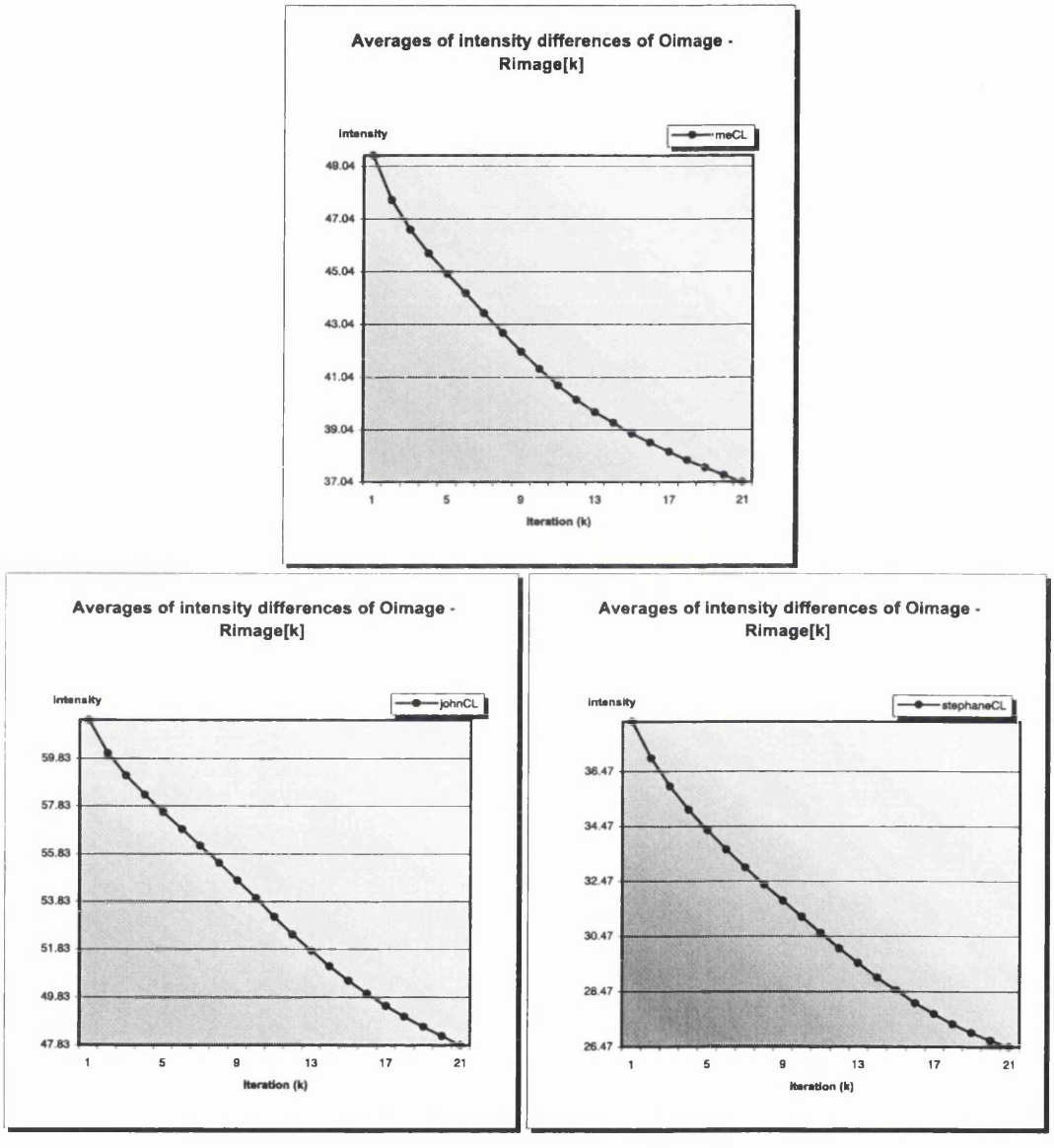


Figure E.3: *AVID* charts for *Album2*

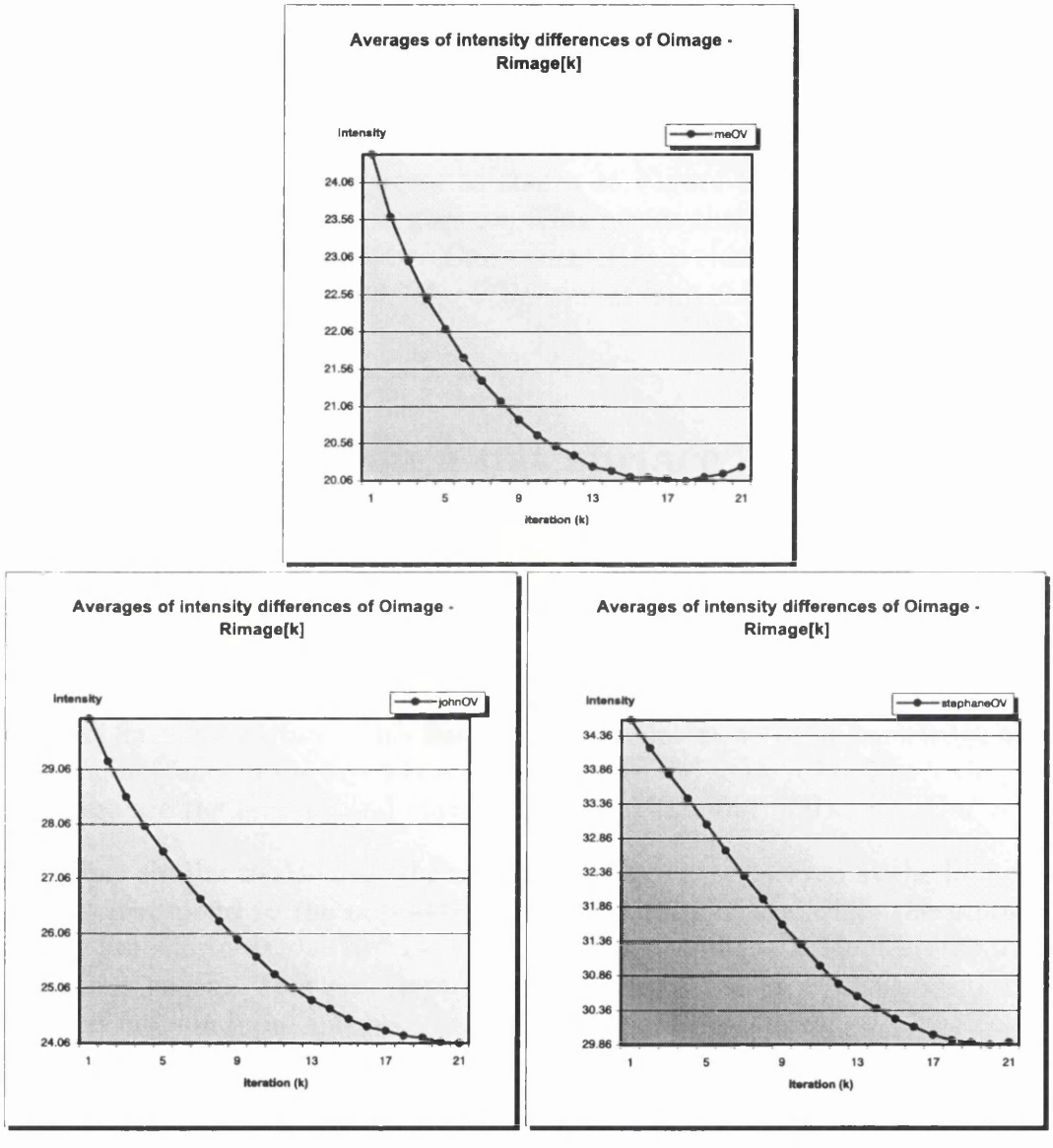


Figure E.4: *AVID* charts for a half ovoid

Consequently, average intensity errors for *Album2* decreased, so our reconstruction method makes a half-cylindrical face approach the original face in each face image.

3.3 Standard deviations of intensity erros: SDID

The curves in Figure E.5 were created by the *SDID* values in Table E.3. They are similar to the *AVID* curves as shown in Figure E.3. They drop linearly as the reconstruction process goes on. This means that the standard deviations from each *AVID* value decrease. Consequently, it is clear that our reconstruction method is stable for *Album2*. The *SDID* curves for a half ovoid can also be found in Figure E.6.

4 Album3 : from a flat surface

4.1 Table of averages and standard deviations of intensity differences

Table E.5 shows the *AVID* and *SDID* values observed in the reconstruction process for a flat surface. This flat surface has almost no prior knowledge about the original faces in the input face images, shown in Figure 6.12. The highlighted numbers are the minima and maxima of the *AVID*s and *SDIV*s for *Album3*.

Being similar to *Album2*, the maximum values are observed at the first rows, which correspond to the impoverished faces given initially, while the minimum values are shown at the 20^{th} iteration. The values in the table decrease as the iterations go on. This also implies that reconstructed faces converge into the original faces in input images.

Generally speaking, the minimum values of *Album3* are larger than the maxima of those for *Album2*. This indicates that a cylindrical surface has more knowledge than a flat surface, and this implies that human faces are more cylindrical than flat.

4.2 Averages of intensity errors: AVID

The curves in Figure E.7 were obtained from the *AVID* values in Table E.5. On the whole, they linearly decrease on the whole as the process goes on. This

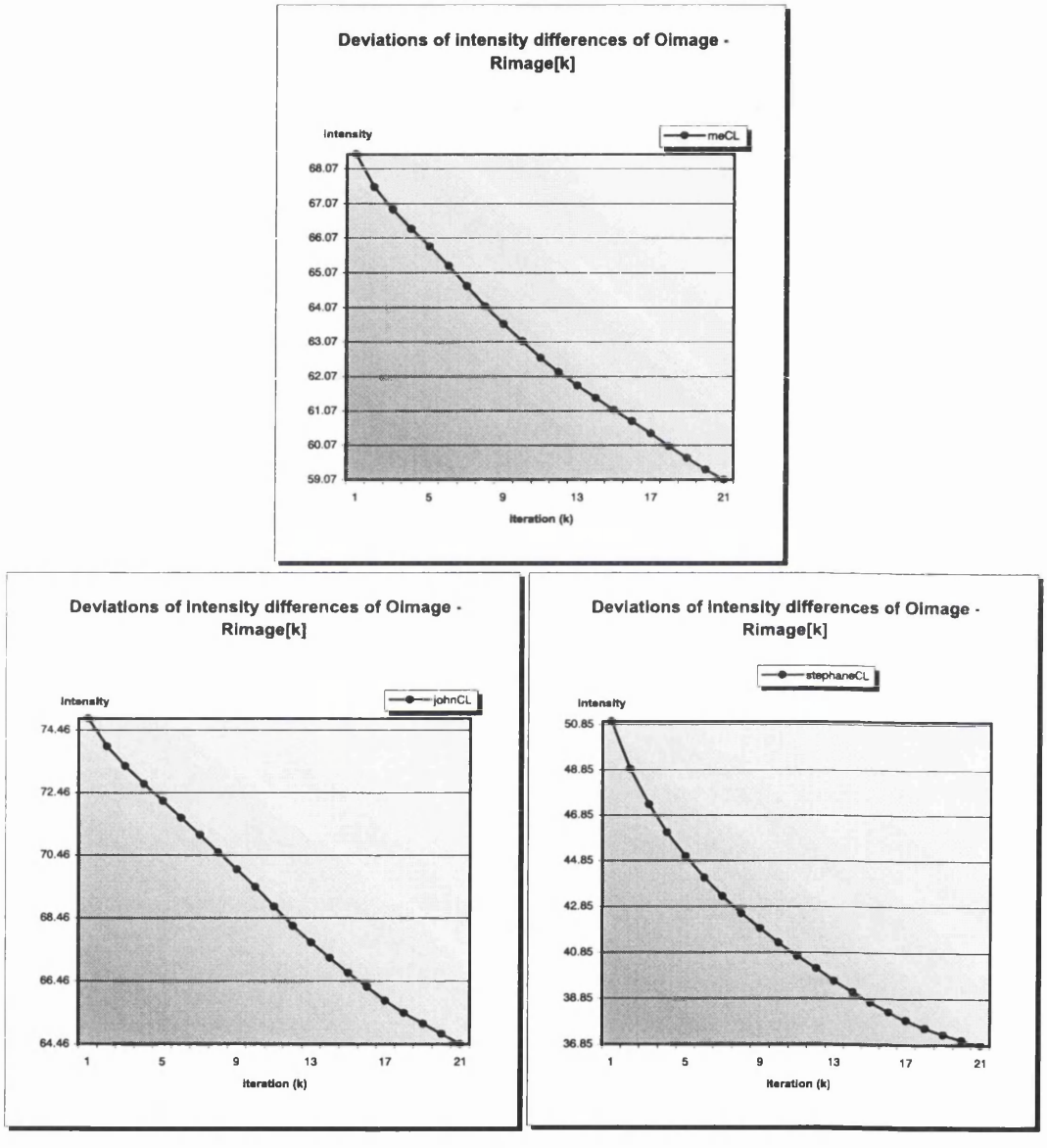


Figure E.5: *SDID* charts for *Album2*

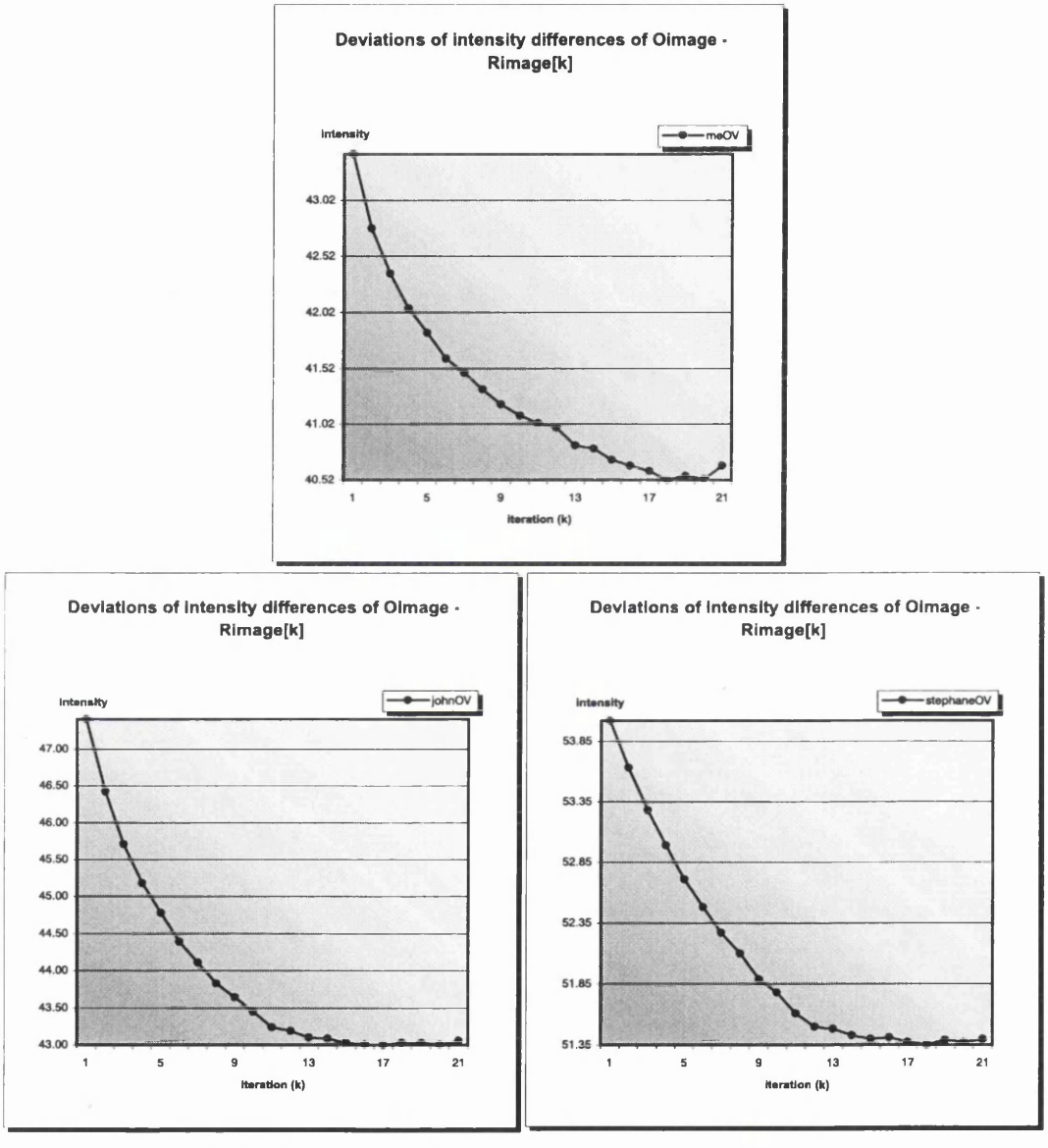


Figure E.6: *SDID* charts for a half ovoid

	Gimage - Rimage[k]							
	Lena		Mozart		mePhoto		yhPhoto	
K	Average	Deviation	Average	Deviation	Average	Deviation	Average	Deviation
0	**78.57**	**46.59**	**105.36**	**73.32**	**60.96**	**46.52**	**59.49**	**51.12**
1	76.54	45.36	100.05	72.17	58.62	46.31	57.28	51.07
2	75.15	44.76	96.79	71.38	57.25	46.26	56.11	51.02
3	74.05	44.40	94.29	70.67	56.14	46.21	55.20	50.96
4	73.11	44.13	92.20	70.05	55.22	46.16	54.46	50.88
5	72.27	43.91	90.38	69.45	54.41	46.13	53.81	50.81
6	71.49	43.71	88.74	68.87	53.71	46.11	53.24	50.74
7	70.76	43.52	87.22	68.31	53.08	46.10	52.71	50.67
8	70.04	43.33	85.77	67.75	52.52	46.09	52.21	50.61
9	69.30	43.16	84.33	67.17	52.02	46.09	51.75	50.54
10	68.54	42.99	82.89	66.59	51.57	46.08	51.29	50.48
11	67.73	42.83	81.49	66.03	51.17	46.07	50.84	50.42
12	66.87	42.67	80.13	65.47	50.81	46.06	50.40	50.36
13	65.96	42.53	78.81	64.94	50.48	46.04	49.99	50.30
14	65.02	42.38	77.54	64.42	50.18	46.02	49.59	50.24
15	64.07	42.21	76.36	63.90	49.92	46.01	49.22	50.17
16	63.09	42.03	75.23	63.37	49.70	45.99	48.87	50.08
17	62.14	41.87	74.09	62.81	49.52	45.96	48.57	49.96
18	61.19	41.76	72.99	62.24	49.35	45.95	48.31	49.80
19	60.29	41.66	71.90	61.63	49.21	45.94	48.09	49.58
20	**59.46**	**41.59**	**70.86**	**60.99**	**49.09**	**45.92**	**47.87**	**49.15**

Table E.5: Experimental values for *Album3*

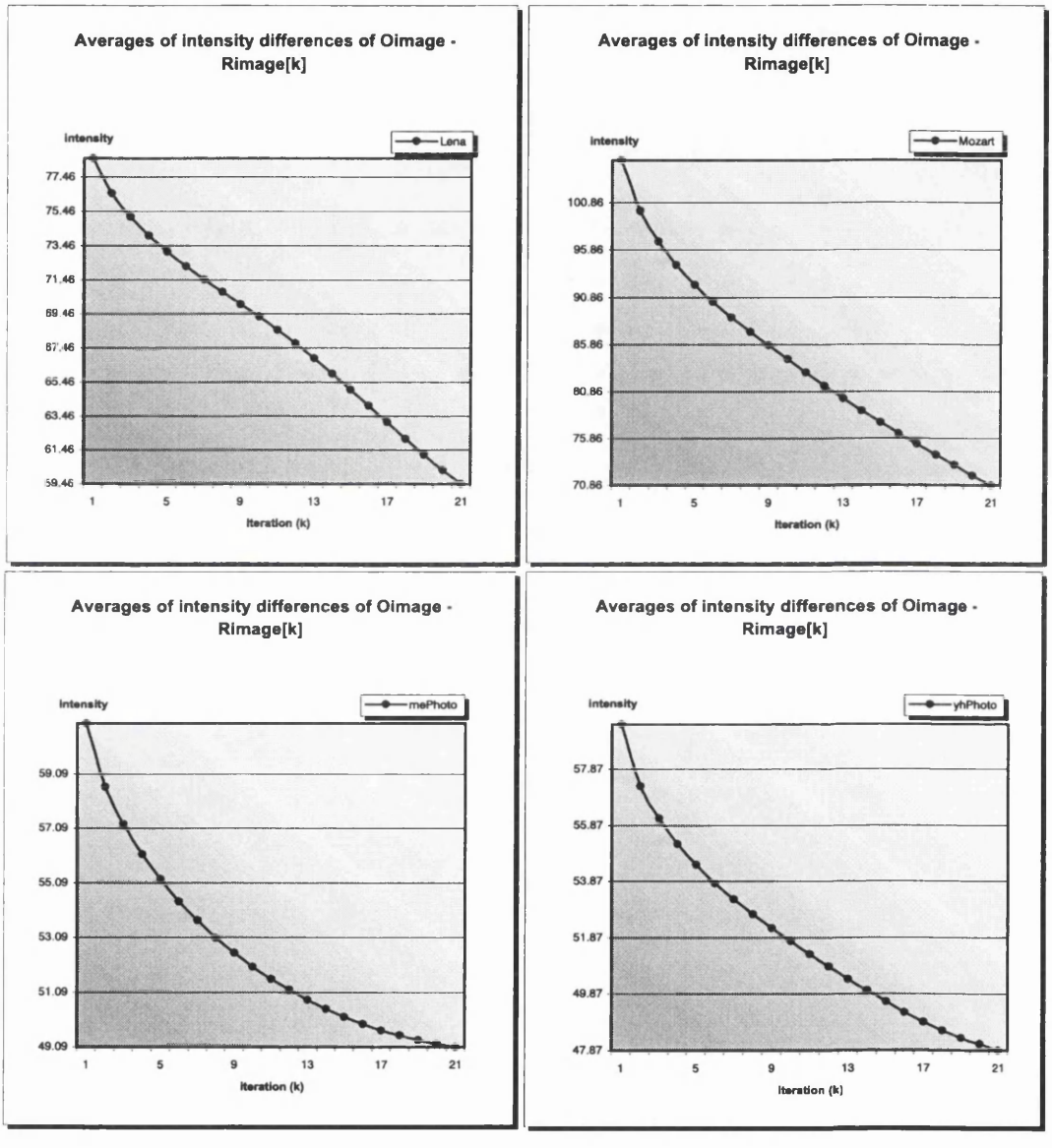


Figure E.7: *AVID* charts for *Album3*

is similar to the curves for *Album2*, because a flat surface also has almost no knowledge about the original faces in the input images in *Album3*.

Average intensity errors, *AVID*s, decrease. Therefore our reconstruction method can produce reconstructed faces that reflect the originals in input face images, starting from a flat surface.

4.3 Standard deviations of intensity erros: SDID

The curves in Figure E.8 were created by the *SDID* values in Table E.5. They decrease as the iteration goes on. That is, the standard deviations from each *AVID* value decrease. Consequently, it is clear that our reconstruction method is stable even when flat faces are applied.

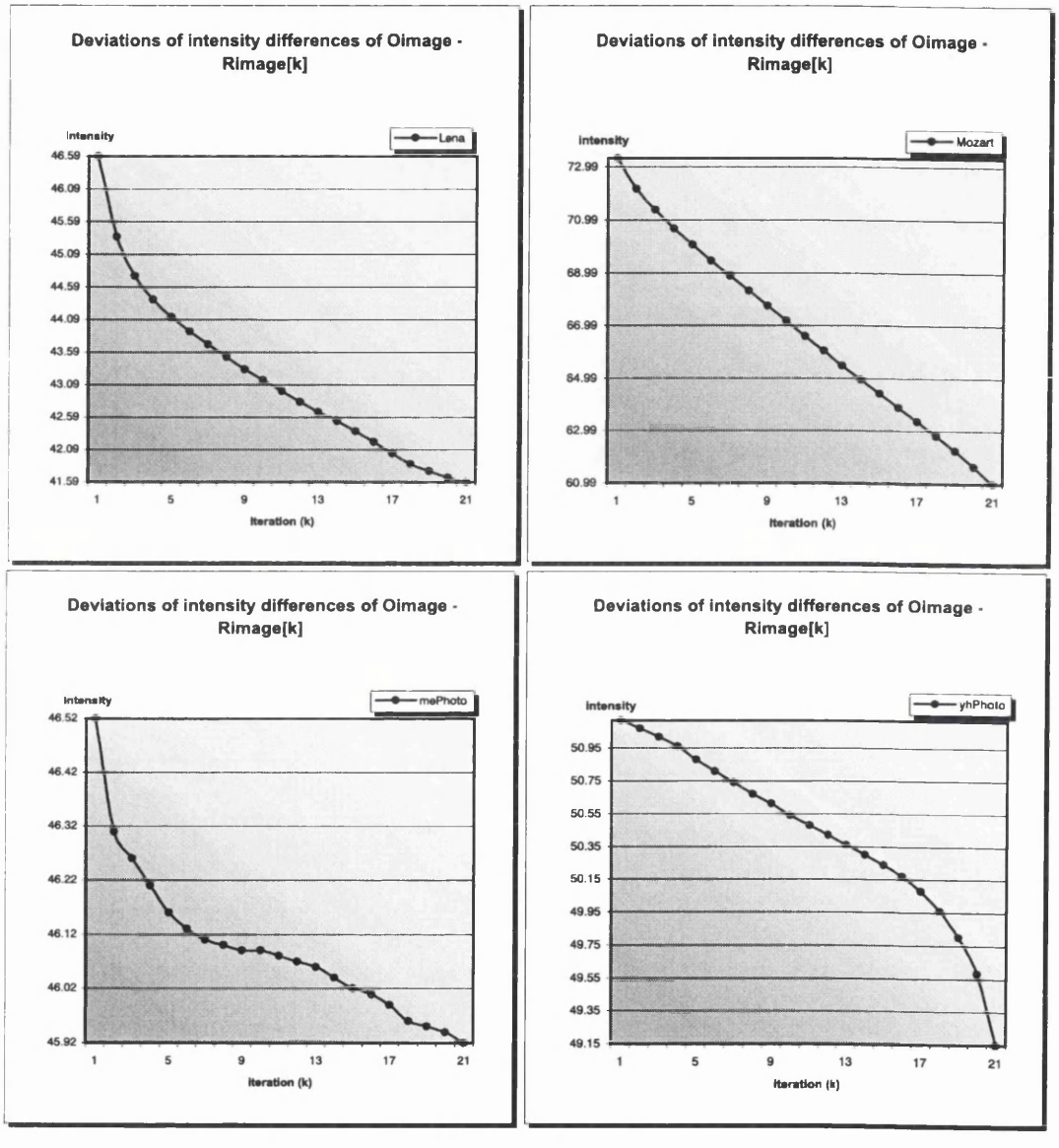


Figure E.8: *SDID* charts for *Album3*

Bibliography

[AC91] Andrew C. Aitchison and Ian Craw. Synthetic images of faces - an approach to model-based face recognition. In Peter Mowforth, editor, *British Machine Vision Conference*, pages 226–232. Springer-Verlag, 1991.

[AGR95] Joseph J. Atick, Paul A. Griffin, and A. Norman Redlich. Statistical approach to shape from shading: Reconstruction of 3d face surfaces from single 2d images. Technical report, Computational Neuroscience Laboratory, The Rockefeller University, NY, 1995.

[AGR96] Joseph J. Atick, Paul A. Griffin, and A. Norman Redlich. The vocabulary of shape: Principal shapes for probing perception and neural response. *Network: Computation in Neural Systems*, 7:1–5, 1996.

[AS96] Maria-Elena Algorri and Francis Schmitt. Mesh simplification. *Computer Graphics Forum*, 15(3):C–77–C–86, Aug 1996.

[ASW93] Takaaki Akimoto, Yasuhito Suenaga, and Richard S. Wallace. Automatic creation of 3d facial models. *IEEE Computer Graphics and Applications*, pages 16–22, sep 1993.

[BB93] Vicki Bruce and Mike Burton, editors. *Processing Images of Faces.* Ablex Publishing Co., 1993.

[BCGH92] Alan H. Barr, Bena Curin, Steven Gabriel, and John F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Computer Graphics Proceedings*, Annual Conference Series, pages 313–320, Chicago, jul 1992. ACM SIGGRAPH.

[BET94] H. H. Bülthoff, S. Y. Edelman, and M. J. Tarr. How are three-dimensional objects represented in the brain? Technical

Report 5, Max Plnack Institut fr biologische Kybernetik, Max-Planck-Institut für biologische Kybernetik, Spemannstr. 38, 72076 Tübingen, Germany, mar 1994.

[BG92] Peter Burger and Duncan Gillies. *Interactive Computer Graphics Functional, Procedural and Device-level Methods.* Addison-Wesley Publishing Company, 1992.

[BH85] Michael J. Brooks and Berthold K. P. Horn. Shape and source from shading. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 932–936. Morgan Kaufmann Publishers, aug 1985.

[Bli77] J. F. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics '77*, 11(2):192–198, jul 1977.

[BN76] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, oct 1976.

[Bou70] W. J. Bouknight. A procedure for the generation of three-dimensional half-toned computer graphics presentations. *Communications of the ACM*, 13(9):527–536, 1970.

[BP92a] Philip J. Benson and David I. Perrett. Face to face with the perfect image. *New Scientist*, pages 32–35, feb 1992.

[BP92b] M. Bichsel and A. P. Pentland. A simple algorithm for shape from shading. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 459–465, Champaign, Illinois, USA, jun 1992. IEEE Computer Society Press.

[BP93] Roberto Brunelli and Tomaso Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, oct 1993.

[Bru82] Anna R. Bruss. The eikonal equation: Some results applicable to computer vision. *Journal of Mathematical Physics*, 23:890–896, may 1982.

[Bru88] Alfred M. Bruckstein. On shape from shading. *Computer Vision, Graphics, and Image Processing*, 44:139–154, 1988.

[BS63] P. Beckmann and A. Spizzochino. *The Scattering of Electromagnetic Waves from Rough Surfaces.* Pergamon, 1963.

[BWR68] D. Buhl, W. J. Welch, and D. G. Rea. Reradiation and thermal emission from illuminated craters on the lunar surface. *Journal of Geophysical Research*, 73(16):5281–5295, aug 1968.

[CCMS97] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 1997(13):228–246, 1997.

[CEL87] I. Craw, H. Ellis, and J. R. Lishman. Automatic extraction of face-features. *Pattern Recognition Letters*, 5:183–187, 1987.

[CG85] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: A radiosity solution for complex environment. In Brian A. Barsky, editor, *Computer Graphics*, volume 19(3), pages 31–40. ACM SIGGRAPH, jul 1985.

[Cla95] Roger J. Clarke. *Digital Compression of Still Images and Video.* Academic Press, 1995.

[CS94] X. Chen and F. Schmitt. Surface modelling of range data by constrained triangulation. *Computer-Aided Design*, 26(8):632–645, aug 1994.

[CSO97] S. I. Cho, H. Saito, and S. Ozawa. Analytical solution of shape from shading problem. In *British Machine Vision Conference*, pages 51–59, 1997.

[CT82] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, jan 1982.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification envelopes. Technical Report TR96-017, Computer Science, University of North Carolina at Chapel Hill, 1996.

[Dev87] Jay L. Devore. *Probability and Statistics for Engineering and the Sciences.* Brooks/Cole Publishing Company, 1987.

[DO92] Paul Dupuis and John Oliensis. Direct method for reconstructing shape from shading. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 453–458. IEEE Computer Society, IEEE Computer Society Press, 1992.

[DS81] P. Deift and J. Sylvester. Some remarks on the shape from shading problem in computer vision. *Journal of Mathematical Analysis and Applications*, 84(1):235–248, 1981.

[DY88] N. D. Duffy and J. F. S. Yau. Facial image reconstruction and manipulation from measurements obtained using a structured lighting technique. *Pattern Recognition Letters*, 7:239–243, apr 1988.

[EDD+95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. Technical Report 95-01-02, Computer Science and Engineering Department, University of Washington, seattle, WA, USA., 1995.

[EM94] H. Edelsbrunner and E. P. Mucke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, jan 1994.

[Eri96] Carl Erikson. Polygonal simplification: An overview. Technical Report TR96-016, Department of Computer Science, UNC-Chapel Hill, Chapel Hill, NC 27599-3175, USA., 1996.

[FC88] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:439–451, jul 1988.

[FP93] T-P. Fanf and L. A. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics and Applications*, pages 36–47, may 1993.

[FS94] L. H. Finkel and P. Sajda. Constructing visual perception. *American Scientist*, 82:224–237, may 1994.

[FvDF+94] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips. *Introduction to Computer Graphics.* Addison-Wesley Publishing Company, 1994.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics.* Addison-Wesley Publishing Company, 1990.

[FZ91] David Forsyth and Andrew Zisserman. Reflections on shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):671–679, jul 1991.

[Gal78] F. Galton. Composite portraits. *Journal of the Anthropological Institute of Great Britain and Ireland*, 8:132–144, 1878.

[Gla89] A. S. Glassner. An overview of ray tracing. In A. S. Glassner, editor, *An introduction to Ray Tracing*, chapter 1, pages 1–31. Academic Press, 1989.

[Gla90] Andrews S. Glassner, editor. *Graphics Gems.* The Graphics Gems Series. Academic Press, united kingdom edition, 1990.

[Gou71] Henri Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, c-20(6):623–629, 1971.

[Hal86] Roy Hall. A characterization of illumination models and shading techniques. *The Visual Computer*, 1986(2):268–277, 1986.

[Ham93] Bernd Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, apr 1993.

[Hap66] Bruce Hapke. An improved theoritical lunar photometric function. *The Astronotical Journal*, 71(5):333–339, jun 1966.

[HB86] Berthold K. P. Horn and Michael J. Brooks. The variational approach to shape from shading. *Computer Vision, Graphics and Image Processing*, 33(2):174–208, Feburary 1986.

[HB88] Glenn Healey and Thomas O. Binford. Local shape from specularity. *Computer Vision, Graphics and Image Processing*, 42:62–86, 1988.

[HB89] Berthold K. P. Horn and Michael J. Brooks, editors. *Shape from Shading.* The MIT Press, Cambridge, Massachusetts, 1989.

[HB94] Donald Hearn and M. Pauline Baker. *Computer Graphics.* Prentice-Hall International, second edition, 1994.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Computer Graphics Proceedings*, Annual Conference Series, pages 19–26, New York, aug 1993. ACM SIGGRAPH.

[Hec86] P. S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, 1986.

[HH93] Paul Hinker and Charles Hansen. Geometric optimization. In Gregory M. Nielson and Dan Bergeron, editors, *Proceedings of Visualization*, pages 189–195, 1993.

[HL88] K Ho-Le. Finite element mesh generation methods: a review and classification. *Computer-Aided Design*, 20(1):27–38, jan/feb 1988.

[Hop94] Hugues Hoppe. *Surface Reconstruction from Unoranized Points.* PhD thesis, University of Washington, USA, 1994.

[Hor75] Berthold K. P. Horn. Obtaining shape from shading information. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 115–155. McGraw-Hill, New York, 1975.

[Hor81] Berthold K. P. Horn. Hill shading and the reflectance map. *Proceedings of The IEEE*, 69(1):14–47, jan 1981.

[Hor86] Berthold K. P. Horn. *Robot Vision.* The MIT Press, 1986.

[Hor90] Berthold K. P. Horn. Height and gradient from shading. *International Journal of Computer Vision*, 5(1):37–75, 1990.

[HS89] B. K. P. Horn and R. W. Sjoberg. Calculating the reflectance map. In B. K. P. Horn and M. J. Brooks, editors, *Shape from Shading*, pages 215–244. The MIT Press, 1989.

[HTSG91] Xiao D. He, Kenneth E. Torrance, F. X. Sillion, and Donald P. Greenberg. A comprehensive physical model for light reflection. In *Computer Graphics*, volume 25(4) of *Annual Conference Series*, pages 175–186. ACM SIGGRAPH, jul 1991.

[HY96] H.S.Ip Horace and L. Yin. Constructing a 3d individualized head model from two orthogonal views. *The Visual Computer*, 1996(12):254–266, 1996.

[IH81] Katsushi Ikeuchi and Berthold K. P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17(3):141–184, 1981.

[IS91] K. Ikeuchi and K. Sato. Determining reflectance properties of an object using range and brightness images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1139–1153, 1991.

[Jol86] I. T. Jolliffe. *Principal Component Analysis.* Springer Verlag, New York, 1986.

[Kaj85] James T. Kajiya. Anisotropic reflection models. In Brian A. Barsky, editor, *Computer Graphics*, volume 19(3), pages 15–21. ACM SIGGRAPH, jul 1985.

[KGC96] R. M. Koch, M. H. Gross, and F. R. Carls. Simulating facial surgery using finite element models. In *Computer Graphics Proceedings*, Annual Conference Series, pages 421–428. ACM SIGGRAPH, aug 1996.

[Kir92] David Kirk, editor. *Graphics Gems III.* The Graphics Gems Series. Academic Press, united kingdom edition, 1992.

[KS90] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[KT96] Alan D. Kalvin and Russell H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, pages 64–77, may 1996.

[Kum96] S. Kumar. Surface triangulation: A survey. Technical report, Computer Science Department, University of North Carolina, Chapel Hill NC 27599, USA., jul 1996.

[KvD80] J. J. Koenderink and A. J. van Doorn. Photometric invariants related to solid shape. *Optica Acta*, 27(7):981–996, 1980.

[KvD82] J. J. Koenderink and A. J. van Doorn. The shape of smooth objects and the way contours end. *Perception*, 11:129–137, 1982.

[KvD83] J. J. Koenderink and A. J. van Doorn. Geometrical modes as a general method to treat diffuse interreflections in radiometry. *Journal of the Optical Society of America*, 73(6):843–850, 1983.

[KvD84] J. J. Koenderink and A. J. van Doorn. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.

[Lan94] Michael S. Langer. *Shading Computations on the Radiation Manifold.* PhD thesis, McGill University, Montreal, Department of Electrical Engineering, oct 1994.

[LB91] Yvan G. Leclerc and Aaron F. Bobick. The direct computation of height from shading. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 552–558. IEEE Computer Society, IEEE Computer Society Press, jun 1991.

[Lew94] Robert R. Lewis. Making shaders more physically plausible. *Computer Graphics forum*, 13(2):109–120, feb 1994.

[Lin93] Alfred D. Linney. The use of 3-d computer graphics for the simulation and prediction of facial surgery. In Vicki Bruce and Mike Burton, editors, *Processing Images of Faces*, chapter 7, pages 149–178. Ablex Publishing Corporation, Norwood, New Jersey, Second Printing 1993.

[LK93] Kyoung Mu. Lee and C.-C. Jay. Kuo. Shape from shading with a linear triangular element surface model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):815–822, aug 1993.

[LR85] Chia-Hoang Lee and Azriel Rosenfeld. Improved methods of estimating shape from shading using the light source coordinate system. *Artificial Intelligence*, 26:125–143, 1985.

[LTW93] Y. Lee, D. Terzopoulos, and K. Waters. Constructing physics-based facial models of individuals. In *Proceedings Graphics Interface '93*, pages 1–8. Canadian Information Processing Society, may 1993.

[Maz94] D. Mazzone. *Sculpturing.* Walter Foster Publishing, 1994.

[MC96] S. D. Michael and M. Chen. The 3d reconstruction of facial features using volume distortion. In H. Jones, R. Raby, and D. Vicars, editors, *Eurographics UK chapter 14th Annual Conference*, volume 2, pages 297–305, Imperial Colledge, London, mar 1996.

[MCvdM92] B. S. Manjunath, R. Chellappa, and C. von der Malsburg. A feature based approach to face recognition. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 373–378, Champaign, Illinois, USA, jun 1992. IEEE Computer Society, IEEE Computer Society Press.

[MJDZ91] Jr. M. J. DeHaemer and M. J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers & Graphics*, 15(2):175–184, 1991.

[Muk95] M. Mukerjee. About face. Scientific American, dec 1995.

[Mur91] S. Muraki. Volumetric shape description of range data using blobby model. In *Computer Graphics*, volume 25(4) of *Annual Conference Series*, pages 227–235. ACM SIGGRAPH, jul 1991.

[Mv96] J. D. Murray and W. vanRyper. *Encyclopedia of Graphics File Formats.* O'Reilly & Associates, Inc, second edition, 1996.

[Nal93] Vishvjit S. Nalwa. *A Guided Tour of Computer Vision.* Addison Wesley, 1993.

[NHRD90] M. Nahas, H. Huitric, M. Rioux, and J. Domey. Facial image synthesis using skin texture recording. *The Visual Computer*, 1990(6):337–343, 1990.

[NIK90] S. K. Nayar, K. Ikeuchi, and T. Kanade. Determining shape and reflectance of hybrid surfaces by photometric sampling. *IEEE Transactions on Robotics and Automation*, 6(4):418–431, 1990.

[NIK91] S. K. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: Physical and geometrical perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):611–634, 1991.

[OD93] John Oliensis and Paul Dupuis. A global algorithm for shape from shading. In *Proceedings of the 4th International Conference on Computer Vision*, pages 692–701, Berlin, Germany, 1993. IEEE.

[Oli91a] J. Oliensis. Uniqueness in shape from shading. *International Journal of Computer Vision*, 6(2):75–104, 1991.

[Oli91b] John Oliensis. Shape from shading as a partially well-constrained problem. *Computer Vision Graphics, Image Process(CVGIP): Image Understanding*, 54(2):163–183, sep 1991.

[ON94] Michael Oren and Shree K. Nayar. Generalization of lambert's reflectance model. In *Computer Graphics Proceedings*, Annual Conference Series, pages 239–246. ACM SIGGRAPH, jul 1994.

[OP96] K. M. Oh and K. H. Park. A type-merging algorithm for extracting an isosurface from volumetric data. *The Visual Computer*, 1996(12):406–419, 1996.

[OVBT95] A. J. O'Toole, T. Vetter, H. H. Bulthoff, and N. F. Troje. The role of shape and texture information in sex classification. Technical Report 23, Max Planck Institut fur biologische Kybernetik, Tubingen, Germany, dec 1995.

[Par82] Frederic I. Parke. Parameterized models for facial animation. *IEEE Computer Graphics and Applications*, pages 61–68, nov 1982.

[Par91] R. Parslow. Why is everyone 3-d blind? In F. H. Post and W. Barth, editors, *EUROGRAPHICS'91*, pages 367–370, North-Holland, Amsterdam, sep 1991. Eurographics association, Elsevier Science Publishers B. V.

[Pen82] Alex P. Pentland. Finding the illuminant direction. *Journal of the Optical Society of America*, 72(4):448–455, 1982.

[Pen86] Alex P. Pentland. Local shading analysis. In Alex P. Pentland, editor, *From Pixels to Predicates*, pages 40–77. Ablex Publishing, Norwood, NJ, 1986.

[Pen90] Alex P. Pentland. Linear shape from shading. *International Journal of Computer Vision*, 4:153–162, 1990.

[PF90] Pierre Poulin and Alain Fournier. A model for anisotropic reflection. In *Computer Graphics*, volume 24(4), pages 273–282. ACM SIGGRAPH, aug 1990.

[Pho75] Buit-Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, jun 1975.

[Ple89] Daniel Pletinckx. Quaternion calculus as a basic tool in computer graphics. *Visual Computer*, 5:2–13, 1989.

[PTK85] Tomaso Poggio, Vincent Torre, and Christof Koch. Computational vision and regularization theory. *Nature*, 317(26):314–319, sep 1985.

[PW91] Manjula Patel and Philip J. Willis. Faces: Facial animation, construction and editing system. In F. H. Post and W. Barth, editors, *EUROGRAPHICS'91*, pages 33–45, North-Holland, Amsterdam, sep 1991. Eurographics association, Elsevier Science Publishers B. V.

[PW96] F. I. Parke and K. Waters. *Computer Facial Animation.* A K Peters, 1996.

[Ram88a] V. S. Ramachandran. Perception of shape from shading. *Nature*, 331(6152):133–166, jan 1988.

[Ram88b] Vilayanur S. Ramachandran. Perceiving shape from shading. *Scientific American*, pages 58–65, aug 1988.

[Rat72] Floyd Ratliff. Contour and contrast. *Scientific American*, 226(6):91–101, jun 1972.

[RO96] Kevin J. Renze and James H. Oliver. Generalized unstructured decimation. *IEEE Computer Graphics and Applications*, pages 24–32, nov 1996.

[RR96] Remi Ronfard and Jarek Rossignae. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):C–67–C–76, Aug 1996.

[SBD86] Francis J. M. Schmitt, Brian A. Barsky, and Wen-Hui Du. An adaptive subdivision method for surface-fitting from sampled data. In David C. Evans and Russell J. Athay, editors, *Computer Graphics Proceedings*, volume 20(4) of *Annual Conference Series*, pages 179–188, Dallas, aug 1986. ACM SIGGRAPH.

[Sch94] Christophe Schlick. A survey of shading and reflectance models. *Computer Graphics forum*, 13(2):121–131, feb 1994.

[Sho85] Ken Shoemake. Animating rotation with quaternion curves. In Brian A. Barsky, editor, *Computer Graphics Proceedings*, volume 19(3) of *A quarterly report of ACM SIGGRAPH*, pages 245–254, San Francisco, jul 1985. ACM SIGGRAPH.

[Sho87] Ken Shoemake. Quaternion calculus and fast animation. SIGGRAPH'87 Course 10:3D Motion Specification and Control, 1987.

[SK87] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, mar 1987.

[Spe96] Bob Spence. Visualisation really has nothing to do with computers. In H. Jones, R. Raby, and D. Vicars, editors, *Eurographics UK Chapter 14th Annual Conference*, pages 1–8, Imperial College, London, mar 1996. Eurographics association.

[Ste95] S. V. Stevenage. Expertise and the caricature advantage. In Tim Valentine, editor, *Cognitive and Computational Aspects of Face Recognition: Explorations in face space*, chapter 2, pages 24–46. Routledge, 1995.

[SU94] J. P. Siebert and C. W. Urquhart. C3d: a novel vision-based 3-d data acquisition system. The Turing Institute, 77-81 Dumbarton Road, Glasgow G11 6PP Scotland, nov 1994.

[Sze91] Richard Szelisk. Fast shape from shading. *Computer Vision Graphics, Image Process(CVGIP): Image Understanding*, 53(2):129–153, 1991.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *Computer Graphics Proceedings*, Annual Conference Series, pages 65–70, Chicago, jul 1992. ACM SIGGRAPH.

[Tho80] Peter Thompson. Margaret thatcher: a new illusion. *Perception*, 9:483–484, 1980.

[TK92] H. T. Tanaka and F. Kishino. Recovering and visualizing complex shapes from range data. In T. L. Kunii, editor, *Visual Computing integrating Computer Grapgics with Computer Vision*, chapter 6, pages 331–348. Springer-Verlag, 1992.

[TS67] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 57(9):1105–1114, 1967.

[Tur92] Greg Turk. Re-tiling polygonal surfaces. In *Computer Graphics Proceedings*, Annual Conference Series, pages 55–64, Chicago, jul 1992. ACM SIGGRAPH.

[TV91] D. Terzopoulos and M. Vasilescu. Sampling and reconstruction with adaptive meshes. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 70–75. IEEE Computer Society, IEEE Computer Society Press, 1991.

[Val95] Tim Valentine, editor. *Cognitive and Computational Aspects of Face Recognition: Explorations in face space.* Routledge, 1995.

[Var94] Amitabh Varshney. *Hierarchical Geometric Approximations.* PhD thesis, University of North Carolina, Chapel Hill, USA, 1994.

[Vet96] Thomas Vetter. Synthesis of novel views from a single face image. Technical Report 26, Max Planck Institut, Biologische Kybernetik, Tubingen, Germany, feb 1996.

[VY93] Omar E. Vega and Y. H. Yang. Shading logic: A heuristic approach to recover shape from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):592–597, jun 1993.

[Wat87] Keith Waters. A muscle model for animating three-dimensional facial expression. In Maureen C. Stone, editor, *Computer Graphics Proceedings*, volume 21(4), pages 17–24, New York, jul 1987. ACM SIGGRAPH.

[Wat94] Alan Watt. *3D Computer Graphics.* Addition-Wesley, second edition, 1994.

[WH97] P. L. Worthington and E. R. Hancock. Needle map recovery using robust regularizers. In *British Machine Vision Conference*, pages 31–40, 1997.

[Whi80] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, jun 1980.

[Wil90] Lance Williams. 3d paint. In *Computer Graphics Proceedings*, number 2 in 24, pages 225–233, Snowbird, Utah, mar 1990. ACM SIGGRAPH.

[Wol92] Lawrence B. Wolff. Diffuse reflection. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 472–478, Champaign, Illinois, USA, jun 1992. IEEE Computer Society, IEEE Computer Society Press.

[Woo77] Robert J. Woodham. A cooperative algorithm for determining surface orientation from a single view. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2, pages 635–641, aug 1977.

[Woo80] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[Woo81] Robert J. Woodham. Analysing images of curved surfaces. *Artificial Intelligence*, 17:117–140, 1981.

[Woo85] Tony C. Woo. A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics and Applications*, pages 19–27, mar 1985.

[WT91] K. Waters and D. Terzopoulos. Modelling and animating faces using scanned data. *The Journal of Visualization and Computer Animation*, 2:123–128, jun 1991.

[WW93] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques theory and Practice.* Addition-Wesley, 1993.

[YD88] John F. S. Yau and Neil D. Duffy. A texture mapping approach to 3-d facial image synthesis. *Computer Graphics forum*, 7(1988):129–134, 1988.

[Yin69] R. K. Yin. Looking at up-side-down faces. *Journal of Experimental Psychology*, 81:141–145, 1969.

[ZC91] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo, and shape from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):680–702, jul 1991.

[ZTCS94] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Analysis of shape from shading techniques. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 377–384. IEEE Computer Society Press, jun 1994.

Index