THE MODELLING OF NATURAL IMPERFECTIONS AND AN IMPROVED SPACE FILLING CURVE HALFTONING TECHNIQUE

BY

TIEN-TSIN WONG

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF PHILOSOPHY

DIVISION OF COMPUTER SCIENCE

THE CHINESE UNIVERSITY OF HONG HONG

JUNE 1994

Acknowledgement

I would like to express my deepest gratitude to my dissertation advisor and friend Dr. S. C. Hsu for his guidance and help throughout this dissertation. I would like to extend my thanks to my friends, Chung-yuen Li and Sau-yuen Wong for their valuable comments and help throughout the project. I would like to thank Chung-yuen Li for his help in the preparation of the photographs used in this thesis, and Chi-lok Chan for his help throughout the preparation of this thesis.

I would like to extend my thanks to the Chinese University of Hong Kong for the two years financial support in forms of teaching assistantship and research assistantship.

During my six years study in the Chinese University, I have encountered numerous people who have helped me in many other ways and to whom I am thankful for.

Finally, this dissertation is dedicated to the ones who are always in my heart.

Abstract

In this thesis, we tackle two computer graphics problems: 1) The modelling of natural imperfections and, 2) Improvement on the clustered-dot space filling curve halftoning method.

The modelling of natural imperfections is a technique to introduce blemishes onto the object surfaces in order to achieve the goal of realism. In this project, we present a new framework of modelling of such surface imperfections. The framework consists of two phases: Firstly, a tendency value, which represents the potential of a surface point of interest being imperfect (e.g. with scratches, rusted, etc.), is calculated based on the object orientation to all *abstract imperfection sources.* It is then adjusted by the external factors, like surface exposure, scraping and surface curvature, etc. Secondly, a chaotic imperfection pattern is generated according to this calculated tendency. We have applied this framework to model two very common kinds of imperfections: dust accumulation and scratching. Some promising images are resulted.

Two common problems exist in most of the existing digital halftoning techniques. They are the presence of annoying visual artifacts and ink smudging which introduces intensity error to the final appearance of the halftone. In 1991, a clustered-dot space filling curve halftoning technique was proposed. It reduces both problems effectively. However, a new problem arises. It suffers from excessive blurring when the cluster size increases. In this thesis, we propose two improvements, *selective precipitation* and *adaptive clustering*, on the

former method to minimize these blurring effects. Selective precipitation outputs the sequence of black dots at the position where the sum of the gray values of the corresponding pixels on the original image is the highest. Adaptive clustering uses a 1D edge detection filter to find the sharp edges and adaptively adjust the size of the clusters so that they would not cross over the sharp edges. Hence, the sharpness of fine details is retained without the need of image prefiltering. By calculating a Gibbs measure and the length of the perimeter of the resulting black pixels as objective image quality measures, we have shown that images dithered by our methods do have better quality than the one produced by the original clustered-dot space filling curve halftoning technique.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the course of this research, two different topics in computer graphics are tackled. They are the digital halftoning with space filling curve and the modelling of natural imperfections.

1.1 The Modelling of Natural Imperfections

One can easily synthesize realistic image of perfectly clean objects using state of the art computer graphics technologies. However, objects in reality are seldom free from blemishes. Surfaces in real life are usually covered with different kinds of imperfections such as dust, scratches, rust, splotches, mold and stains.

To achieve the goal of photo-realism, the modelling of natural imperfection clearly cannot be neglected. This is especially important in film production, since the computer generated objects are usually overlaid with live-action scenes [ROBE93]. Surfaces without blemishes make a scene looks unnatural and dissonant.

We state some criteria in order to make a modelling technique useful:

- The technique should simulate the phenomenon realistically.

- The technique should provide sufficient control for the user to design the effect that he/she desires.
- The technique should take care of all small details automatically, leaving the user to concentrate on the design of global appearance.
- The technique should be efficient enough to prevent the rendering process from being slowed down.
- The technique should be compatible with traditional rendering techniques, so that, it can be easily plugged into existing rendering software.

Unfortunately, most current technologies do not satisfy all of the above criteria. In this project, we have developed an empirical framework to simulate the appearance of natural imperfections. We have applied the framework to model dust accumulation and scratching. Although we have only modelled these two kinds of imperfections, our methods can be extended to simulate other imperfections. We show that our techniques satisfy all criteria stated before.

1.2 Improved Clustered-dot Space Filling Curve Halftoning Technique

Digital halftoning is a technique to simulate images with more shades using a limited number of colors. It is especially important in publishing industry, since it is more economic to use fewer colors.

Many halftoning methods have been proposed in the past. Nevertheless, two common problems exist among these methods. Firstly, some halftoning methods produce distracting regular patterns. Secondly, some halftoning methods produce dispersed-dot dithered images, which tends to be darkened excessively when printing smudge exists. Since no printing device is perfect, printing smudge is almost unavoidable in any printing process.

In 1982, a new approach to dithering was proposed. It dithers images along a fractal space filling curve [WITT82]. The intertwined behavior of space filling curve reduces the noticeable regular patterns in the dithered image. In 1991, a clustered-dot version [VELH91] was proposed which also reduces the printing smudge error. It seems that major problems have been solved, at least partially. However, new problem arises. This clustered-dot version suffers from excessively blurring when the cluster size increases (We shall explain the term *cluster size* in chapter 3).

We are proposing an improved method aiming at reducing the blurring appearance while preserving advantages of the original method.

1.3 Structure of the Thesis

Chapter 2 describes our newly proposed framework for imperfection modelling in detail. Two kinds of imperfections, dust accumulation and scratching are modelled using the proposed framework. In chapter 3, we first discuss the problems of existing halftoning methods. Then, we show how our improvements can reduce the blurring problem. Finally, in chapter 4, some conclusions on these two topics are drawn and future works are discussed.

Chapter 2

The Modelling of Natural Imperfections

2.1 Introduction

Although state-of-the-art computer graphics technologies allow us to synthesize realistic images of virtual world, one can easily distinguish synthetic images from photographs of real world in most of the cases. It shows that synthetic images have some characteristics in common which make them look artificial. One such characteristics is the cleanliness of synthetic object surfaces. Synthetic objects are usually perfectly smooth, neat and clean (Figure 2.1). On the other hand, real world objects are usually covered with different kinds of blemishes. These blemishes include dust (Figure 2.2 and 2.3), scratches (Figure 2.5 and 2.6), rust (Figure 2.5), mould (Figure 2.7) and stains.

The modelling of natural imperfections is a technique to introduce blemishes into images in order to achieve realism. Imperfection is a very general concept. The term *imperfection* used in this thesis is restricted to surface imperfections.

To synthesize realistic images, blemish modelling is apparently not negligible.

Figure 2.1: A synthesized image by Mike Miller. It demonstrates one problem of synthetic images, virtual objects are too clean and perfect.

This modelling technique is especially important when computer generated objects are overlaid with scenes of real world. For instance, in the production of the film *Jurassic Park* [ROBE93], computer generated dinosaurs are frequently overlaid with live-action scenes. Surfaces without blemishes make the final scenes look unnatural and dissonant.

A few techniques (discussed in Section 2.2) are proposed to model the appearance of imperfect surfaces. However, they all fail to recognize the common characteristics of blemish distribution. Although blemishes seem to occur randomly, we can find some geometrical factors that affect the distribution of blemishes. For instance, less frequently exposed surface has higher tendency to accumulate dust than frequently exposed one. Paint on a curved surface has higher tendency of being peeled off than that on a flat one. These geometrical informations can be used to automate the generation of blemished patterns.

In this chapter, we have proposed a new empirical framework for the modelling of various imperfections. Our proposed techniques provide sufficient automation, by utilizing the geometrical information of the object model, for a non-professional person to create realistic blemishes. On the other hand, our

methods also provide sufficient control on the overall blemish distribution.

We have applied this framework to model two common imperfections: dust accumulation and scratching. Dust accumulation occurs almost everywhere. Scratching is easily observed on multi-layered surfaces. Although these two imperfections seem unrelated, our framework has been successfully applied to model them.

2.2 Related Work

2.2.1 Texture Mapping

Texture mapping [BLIN76, HECK86] is a simple and powerful technique to increase the visual richness of object surfaces. Its simplicity and efficiency make it popular. The basic idea of texture mapping is to change surface properties of the object according to a user-defined texture image.

Many variants of texture mapping [HECK86] were proposed during the past three decades. The original texture mapping [CATM74] changes only the color parameter of the surface. If the specular reflection is changed according to the texture, it is known as environment mapping [BLIN76, GREE86]. If the normal vector is perturbed according to the texture, it is known as bump mapping [BLIN78]. Others change the glossiness coefficient [BLIN78], transparency [GARD84, GARD85], diffused reflection [MILL84], surface displacement [UPST89], and local coordinate system [KAJI85, CABR87].

It is natural to use texture mapping to visually simulate the surface imperfection by composing texture with blemished patterns. However, it may require many working hours of a professional artist to compose such texture. Texture mapping alone provides no automation of generation of blemishes. Moreover,

texture are usually distorted after mapped onto undevelopable surface.[1] Hanrahan and Harberli [HANR90] developed a 3D painting system which allows composition of texture on the surface of a 3D object interactively. This approach may reduce the effect of distortion. However, it still provides no automation on generation of blemishes.

2.2.2 Blinn's Dusty Surfaces

In 1982, Blinn [BLIN82] modelled the appearance of dusty surfaces using physical approaches. It concerns the statistical simulation of light passing through and being reflected by clouds of similar small particles, like dust particles (assuming dust particles are similar small spheres). However, he assumed the thickness of the dust layer is constant. He did not provide a method to automatically determine the amount of dust accumulated on surfaces. A constant thickness of dust layer is also unrealistic. Moreover, Blinn has not mentioned the extension of the technique to model any other imperfections.

2.2.3 Imperfection Rule-based Systems

Becket and Badler [BECK90] propose a system dedicated to generation of textures with blemishes. They described two methods for the generation of blemishes on a texture: rule-guided aggregation and 2D fractal subdivision. The system also provides a natural language interface. The user can tell the system where to place blemishes through an English-like language.

Unfortunately, the proposed method does not provide sufficient automation. Just as Blinn's method described before, the system only models the chaotic appearance of blemishes and the user still has to specify where to place such blemishes in detail. The method fails to recognize some common features of the

[1] If a surface is developable, the surface can be unfolded or developed onto a plane without stretching or tearing. For details, see [ROGE90, pp. 420].

distribution of blemishes. We will show, later in this chapter, that the distribution of blemishes is usually related to some geometrical factors. Moreover, specifying the location of blemishes through the natural language interface is clumsy and not interactive. It seems more natural to specify through some interactive programs. Furthermore, adjectives (e. g. good, very good, bad, more and less, etc.) used in this English-like language are usually mapped to some fixed parameter value. This unnecessary discretion and constraint may decrease the flexibility of specifying the continuous parameter values. The texture created using rule-guided aggregation or 2D fractal subdivision is still a two-dimensional texture image. It still has to be mapped onto the object surface. As stated before, the mapping process may introduce distortion if the texture is mapped to an undevelopable surface. Moreover, the system does not model dust accumulation.

2.3 Natural Surface Imperfections

This section describes four commonly observed natural surface imperfections. Photographs of real imperfect surfaces will be shown for illustration. Two of them, dust accumulation and scratching, have been modelled using the new framework proposed in section 2.4.

2.3.1 Dust Accumulation

Dust accumulation takes place everywhere. It can be easily observed in any environment which has not been cleaned for a while. Figure 2.2 and 2.3 shows photographs of a dusty lamp stand and a dusty socket respectively. In Figure 2.2, less inclined surface accumulates more dust than steep one due to gravity and surface friction. The dusty surface looks like a surface covered by a thin layer of fuzzy matters, which is composed of different types of dust particles (Figure 2.4).

Figure 2.3 shows that less exposed surface accumulates more dust than more exposed one due to the sheltering effect. Notice the dust particles surrounding the bumpy characters on the socket in Figure 2.3. In this case, dust particles accumulated on surfaces with less exposure have less chance of being removed by wind or other external forces.

Figure 2.2: Dusty lamp stand.

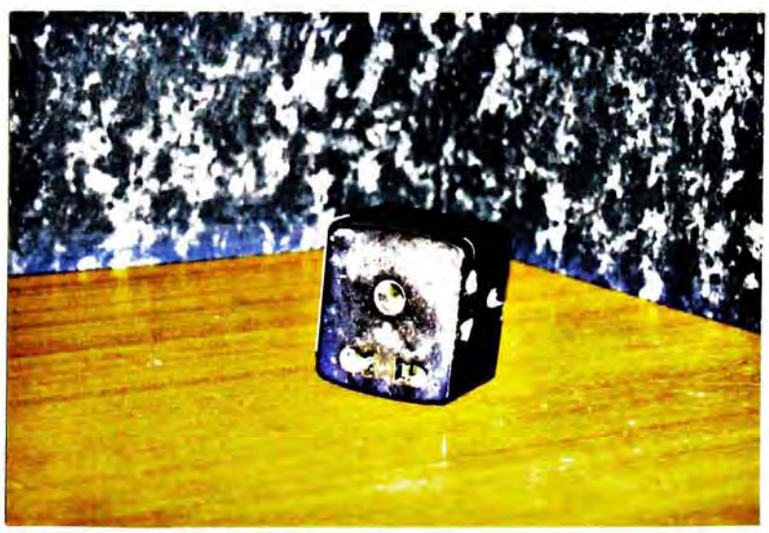

Figure 2.3: Dusty socket.

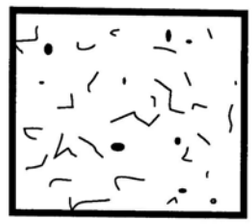

Figure 2.4: A closeup view of dust particles.

2.3.2 Scratching

We define scratching as the phenomenon of parts of the surface layer being peeled off from the object. The occurrence of scratching is usually due to the attack by external forces. This phenomenon is usually seen on the surface of painted object. Actually, it can occur on any multi-layered object, such as gilded objects and leather objects, etc. Figure 2.5 and 2.6 show two different objects with chaotic scratch patterns. Both figures show that scratches usually appear at places with high surface curvature. Surface exposure also affects the formation of scratches, since less exposed surfaces have less chance of being peeled off by any external forces. Notice the leg in the back in Figure 2.5 gets less scratches than the one in the front.

2.3.3 Rusting

Corrosion is the gradual deterioration of any metal due to its reaction with air, water or other chemicals to form oxides or other compounds. Rusting is the term used to describe the corrosion of one specific metal, iron. As iron is the most widely used metal in the world, rusting is the most common type of corrosion. The necessary condition of rusting is the presence of oxygen and water. However, there are some factors catalyzing the rusting process. They are the presence of electrolytes, heat and the presence of another metal, like copper, tin or silver to be in contact with iron. One way to protect iron from

Figure 2.5: Painted chair with scratches.

rusting is to apply a protective layer such as paint, plastic coating, varnish and grease. Therefore, when the protective layer is peeled off from the ferric object, rusting usually occurs. Figure 2.5 shows the reddish brown rust on the scratched portions of the metallic chair. Since rusting usually appears on the scratched surface, curvature and surface exposure affect the tendency of rusting indirectly.

2.3.4 Mould

Mould [To77] is the result of dispersal of fungi on the surface of organic matter, such as bread, cheese, and etc. The appearance of mouldy pattern is usually in the form of aggregates (Figure 2.7) due to the colonization of fungus. Shady,

Figure 2.6: Leather bag with scratches.

wet and warm environment favours the growth of fungi. Again, surface exposure affects the formation of mould. Less exposed surface usually has less chance of being illuminated, hence has a higher tendency of moulding.

Figure 2.7: Bread with mould.

2.4 New Modelling Framework for Natural Imperfections

Under the new framework, the modelling process basically consists of 2 phases: the calculation of tendency value and the generation of chaotic patterns. The first phase involves the determination of a *tendency value* for each surface point of interest. Since the distribution of imperfections is usually related to the surface geometry, we can calculate the tendency value according to these geometrical informations. The tendency value represents the potential of that surface point of interest to become blemished. In the second phase, we generate chaotic patterns according to the calculated tendency. The separation of the modelling process into two phases has an advantage: the generation method of chaotic patterns can be easily replaced with another method to produces different patterns without interferring the tendency calculations.

2.4.1 Calculation of Tendency

Sources of Imperfection

The formation process of blemishes is very complex. It may be due to human factors, physical laws and other unpredictable events. However, we can imagine that the formation of blemishes is due to different kinds of *abstract imperfection sources*. For instance, scratches are usually found near the handle of leather handbag (Figure 2.6). This is due to frequent contact with human hands. In this case, we can imagine that there is an abstract scratch source near the handle. Dust particles are usually accumulating on upward surface due to the effect of one physical law, gravity. We can also imagine that there is a dust source placed above the surface. This abstraction of imperfection sources is analogous to the concept of light sources [FOLE90, ch. 16] [WATT92, pp. 42–48].

It provides control on the overall distribution of blemishes. The user can specify where he/she wants to introduce blemishes by placing the imperfection source nearby. User should find it familiar due to the similarity of this scheme to the light source concept.

We can now calculate a tendency value of a surface point according to its direction and distance to each imperfection source. This tendency has a value between 0 and 1. A surface point with a tendency of value 0 means that it is free of blemishes. A surface point with a tendency of value 1 means that it is certain to be blemished. Consider a tiny surface patch with point P at the center (Figure 2.8), N is the unit normal vector at P, S_i is the i-th imperfection source, U_i is the unit vector pointing from P to S_i, and θ_i is the angle between vector N and U_i. As θ_i decreases, vector N and U_i become closer. The solid angle [2] of the surface patch as seen from S_i increases. Hence the tendency of being affected by the i-th imperfection source increases. Moreover, as the distance ρ_i between S_i and P decreases, the solid angle of the surface patch as seen from S_i increases, hence the tendency increases. Therefore, the tendency due to the i-th imperfection source, T_{S_i} (the subscript S_i stands for the i-th imperfection source), should be inversely related to θ_i and distance ρ_i. The choice of the formula for T_{S_i} depends on the application. We will introduce a useful formulation of T_{S_i} in section 2.5.

Just like the light source, the imperfection source can be in point form or in directional form. Our previous discussion assumes that the source is in point form: i.e. the source is a point in the Euclidean space. It has no volume. This assumption may not be realistic. However, it works fine in practice. Light source in directional form can be regarded as a point source placed at a long distance away. Hence the vector U_i and distance ρ_i is nearly constant for all

[2]Solid angle of a surface element, as seen from a given point, is defined as the projected area of the surface element divided by the square of the distance from the point. See [WATT92, pp. 35–36]

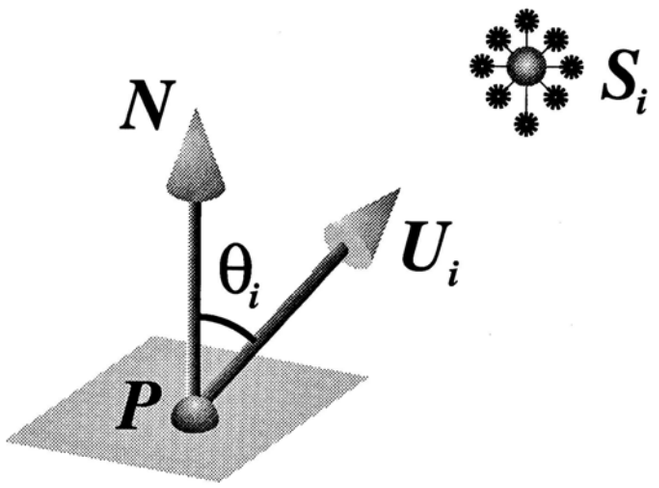

Figure 2.8: A tiny surface patch.

surface points. That is why it is known as directional source. One example of directional source is the Sun. Since the Sun is far away from the Earth, light rays from the Sun are basically in parallel. Since ρ_i is now constant, T_{S_i} is inversely related to θ_i only.

If more than one source is given, we shall sum up the tendency value due to each source (in point form or directional form). Hence, the overall tendency T_S of a surface point due to all sources is

$$T_S = \sum_{i=1}^{m} T_{S_i} \tag{2.1}$$

where m is the total number of imperfection sources and

$$T_{S_i} = \begin{cases} T_{S_i}(\theta_i, \rho_i) & \text{if source is in point form,} \\ T_{S_i}(\theta_i) & \text{if source is in directional form.} \end{cases}$$

External Factors

The tendency value T_S calculated using the above formula is a predicted value. The actual tendency value may deviate from the predicted value due to external

factors. Different kinds of imperfections may be affected by different external factors. Nevertheless, there is one common external factor. It is surface exposure. In this section, we will describe how we can model this geometrical factor. Notice that other external factors also exist and will be discussed when we apply the framework to model dust accumulation and scratching in section 2.5 and 2.6. In general, the actual tendency T for a surface point of interest, after considering those external factors, is the tendency T_S scaled by a function α of all external factors E_i's.

$$T = \max(0, \min(1, T_0 + T_S \cdot \alpha(E_1, E_2, E_3, \ldots))) \tag{2.2}$$

where E_i is the i-th external factor.

T_0 is a constant, s. t. $0 \leq T_0 \leq 1$.

Surface Exposure Consider the intersection of a ray R_i, which is fired from a surface point P, and an obstacle at distance d_i from P (Figure 2.9). As d_i increases (i.e. the obstacle moves away from point P), the effect of that obstacle on the surface exposure ξ at P decreases non-linearly. We define a weight function ω of d_i that returns value in $[0, 1]$ s.t. $\omega(d_1) \leq \omega(d_2)$ if $d_1 \leq d_2$, $\omega(d_i) = 0$ when $d_i = 0$, and $\omega(d_i) \to 1$ when $d_i \to +\infty$ (i.e. no intersection). This function should be non-linear. We define this function as,

$$\omega(d_i) = \frac{d_i}{d_h + d_i} \tag{2.3}$$

where d_h is a user-defined constant called the *half-exposure distance*, which is roughly the average spacing between the point P and other nearby objects that would reduce the exposure at point P to 0.5 (Figure 2.10).

If we fire many rays from point P and evenly distribute them in the upper hemisphere of point P (Figure 2.11), the surface exposure ξ is then defined by the following equation,

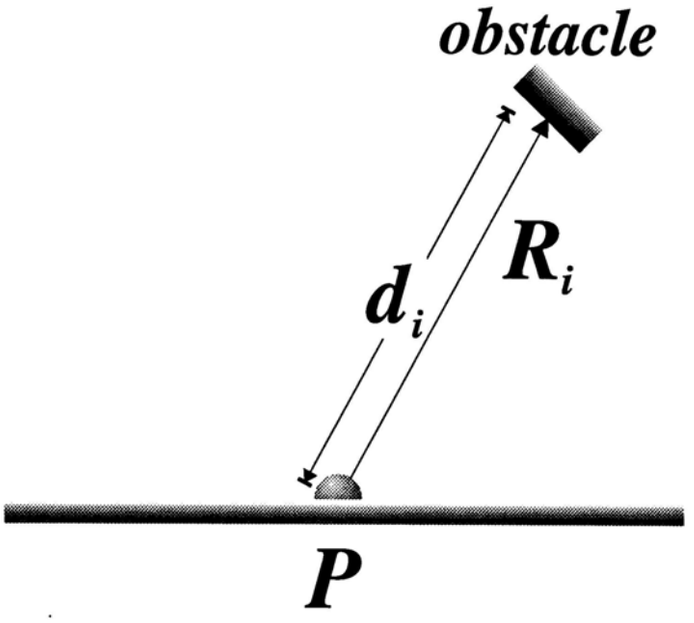

Figure 2.9: Ray R_i intersect with an obstacle.

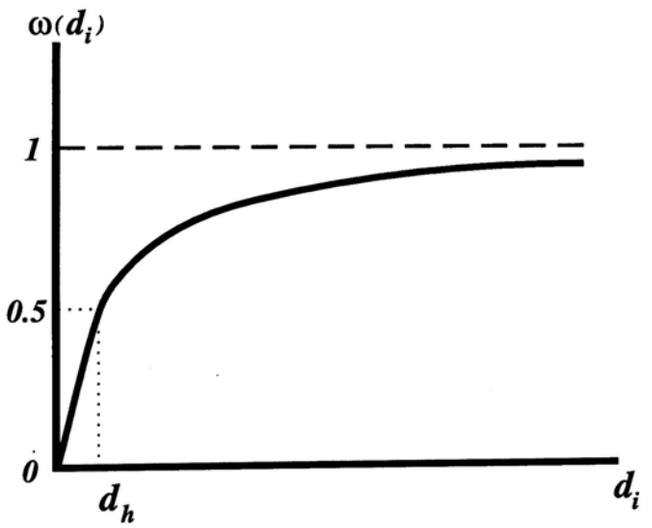

Figure 2.10: The function ω.

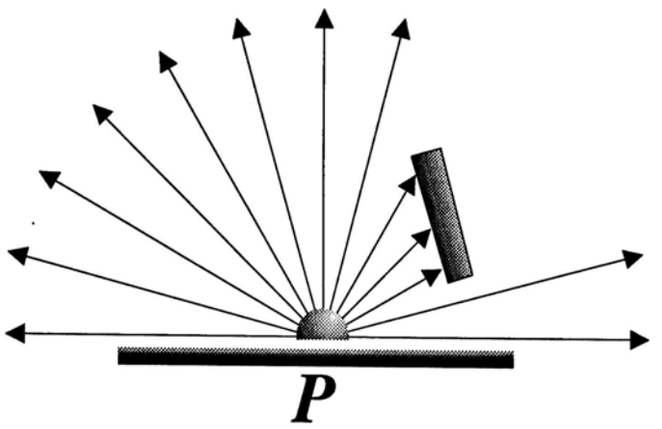

Figure 2.11: n evenly distributed rays emitted from point P.

$$\xi = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \omega(d_i) \tag{2.4}$$

ξ will be 1 when P is completely exposed since all $\omega(d_i) = 1$. ξ will be 0 when P touches with any obstacle since all $\omega(d_i) = 0$.

In practice, we can determine ξ using one-level ray tracing. We subdivide the surface of the upper hemisphere above point P into several equal sized windows. For each window, we fire certain amount of rays from point P through the window to detect whether those rays intersect with any obstacle. Then we can approximate the surface exposure by

$$\xi = \frac{1}{n_0} \sum_{i=1}^{n_0} \omega(d_i) \tag{2.5}$$

where n_0 is number of rays emitted, n_0 is a finite constant,
$\omega()$ and d_i are defined as before.

This is a sampling technique. The more windows are subdivided and the more rays are fired, the more accurate the approximation is. However, this method inherits the same problems as the original ray tracing, namely aliasing

and expensive intersection tests. The aliasing problem can be solved by supersampling [FOLE90, ch. 14], distribution ray tracing [COOK84], cone tracing [AMAN84] or pencil tracing [SHIN87, SHIN89]. However, there is no complete solution to decrease the computational cost of ray tracing to a reasonable level. A more economic approach to determine ξ is described in section 2.5.

2.4.2 Generation of Chaotic Pattern

After the final tendency value T is calculated, we can generate blemishes according to this tendency. Since blemishes are usually in irregular patterns, we need some stochastic methods to generate such patterns. In this section, we suggest two standard methods: solid texturing with fractal/noise function, and rule-guided aggregation.

Solid Texturing with Fractal and Noise Function

Solid texturing [PEAC85, PERL85] is an extension of traditional texture mapping. Traditional texture mapping maps a 2D texture to the surface of an object. This mechanism is altered for solid texture. Instead of 2D texture space, solid texturing uses *3D texture space.* For each point of the object to be textured, there is an associated point in the 3D texture space. The traditional texture mapping is analogous to wrapping the object with a sheet of paper (2D texture space). Solid texturing is analogous to carving the object from a piece of material, like wood and marble (3D texture space). The advantage of solid texturing is that there is no distortion during mapping. The 3D texture can be obtained by creating a 3D lattice of texture values. Mathematical functions can be used to generate values at the points of the 3D lattice. Two popular functions are fractional Brownian motion and Perlin's noise function.

Mandelbrot observed that many natural objects and phenomena are stochastically self-similar [MAND77, PEIT88] which is the main criterion of fractal objects. Since then, fractal has been successfully used in the generation of different natural objects, like landscapes [FOUR82, MAND82, VOSS85, KALR91], plants [OPPE86, KALR91] and clouds [VOSS85]. One of the fractal functions, fractional Brownian motion (fBm) [PEIT88, pp. 58–70], has been used by many researchers [VOSS88, SAUP88, BECK90] to generate interesting chaotic patterns. The original fBm, $V_H(t)$, is a single valued function of one variable, t (usually time). Its increment $V_H(t_2) - V_H(t_1)$ has a Gaussian distribution with variance

$$\langle |V_H(t_2) - V_H(t_1)|^2 \rangle \propto |t_2 - t_1|^{2H}$$

where the brackets $\langle$ and $\rangle$ denote ensemble averages over many samples of $V_H(t)$ and parameter H has a value between 0 and 1. This fBm can be extended to accept positional vector as input. The extended version is frequently used as a solid texturing function. The detailed explanation of fBm can be found in [PEIT88]. We shall not go into detail here.

Perlin [PERL85] proposed another useful stochastic solid texturing function, *noise*(), which is a scalar valued function taking a 3D positional vector as input. It has three major properties:

- Statistical invariance under rotation: it has the same statistical property even when we rotate its domain.
- Statistical invariance under translation: it gives same statistical characteristic no matter how we translate its domain.
- It has no visible features within a certain narrow size range.

The function *noise*() is good because it introduces noise effect without sacrificing the continuity and the control over spatial frequency. Perlin also derived

a turbulence function based on this noise function. He used the noise and turbulence function to generate convincing images of waves and fire.

To generate chaotic patterns, we do the following procedure. Instead of calculating the whole 3D texture space, we only calculate the noise/fBm function whenever necessary. When we shade a point on object surface, we first calculate a noise value using fBm() or noise() with a positional vector as input. The returned noise value is multiplied by the tendency T to create a pattern according to the tendency. Finally, by mapping different ranges of resultant value to different colors, we can receive a chaotic pattern. This method is especially useful in the generation of scratch patterns, rust patterns and fuzzy dust layers.

Rule-guided Aggregation

Becket [BECK90] proposed rule-guided aggregation to generate patterns of stain and rust. Some sticky seeds are first placed on the surface. Then a diffusion process is performed. During this process, whenever a floating particle collides with the sticky seed, there is a chance that the particle will stick to the seed. This possibility is a function of distance between the central seed and the floating particle. Large distance yields small possibility. When the diffusion process finish, aggregation of particles will be formed.

By placing more seeds in the position with a higher tendency value, we can associate the pattern with our calculated tendency value. Notice that this is a method to compose the pattern in the texture space. The mapping process may still introduce distortion.

2.5 Modelling of Dust Accumulation

Dust accumulation is a common natural phenomenon in real life. Modelling of dust accumulation, however, has not been well studied. We present a technique

to model the tendency of dust particles settled on object surface based on the properties of surface and the geometry of object. The scattering of light by airborne dust has been well-studied [BLIN82, MAX86, NISH87, RUSH87] and we shall not consider it here. We illustrate our techniques with images of synthetic dusty objects.

2.5.1 Predicted Tendency of Dust Accumulation

Due to the effect of gravity, horizontal surfaces usually accumulate more dust than inclined ones. However, if an object is placed near an open window in front of a dusty construction site, surfaces facing the window would have more dust particles accumulated on it than on other faces. Therefore inclination of a surface against the direction of the dust source (which might be in terms of the combined effect of gravity, wind direction and the actual source of dust particles) would certainly affect the amount of dust particles settled. On the other hand, this amount is also dependent on the surface type. A sticky surface (a furry, rough or adhesive one) is more likely to accumulate dust particles than less-sticky ones (e.g. a smooth one).

Consider the small surface patch in Figure 2.8. Assuming surface properties are isotropic,[3] the predicted tendency of dust accumulation of a surface point due to a dust source would be a function of θ_i, the angle between surface normal and the vector pointing from P to i-th dust source, and η, a stickiness factor of the surface. Instead of providing a physical model to the dust adhering process, we adopt some functions that approximate the physical process. The function should has its peak when θ_i is 0 and gradually falls off as θ_i increases. Furthermore, an increase in stickiness would decrease the rate of falling off of dust particles. These effects are very similar to the reflection of light from a

[3]We say a surface property is isotropic if the surface property is independent of the viewing direction.

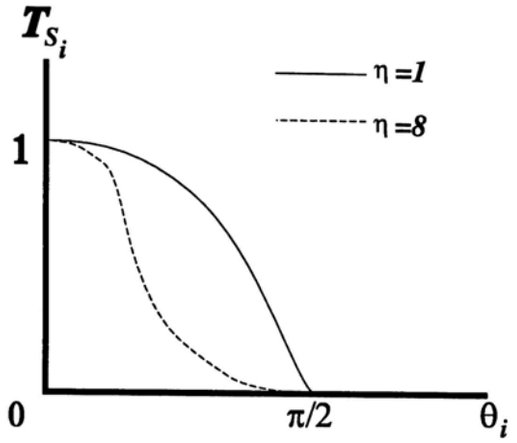

Figure 2.12: Predicted tendency functions with different values of η. ρ_i is kept constant.

surface. The dust source is analogous to the light source. The fall off in dust amount as θ_i increases is similar to the fall off in diffuse component of light reflection. Therefore it is natural to consider a function similar to Phong's reflection model [WATT92] to model this predicted tendency function:

$$T_{S_i} = \begin{cases} \cos^{\eta}(\theta_i) & \text{if source is in directional form,} \\ \frac{1}{\rho_i^2}\cos^{\eta}(\theta_i) & \text{if source is in point form.} \end{cases} \tag{2.6}$$

where θ_i is angle between vector N and U_i,

ρ_i the distance from point P to S_i,

η is the surface slippiness (large η means less sticky).

T_{S_i} has all the required properties (Figure 2.12): A highly sticky surface (solid curve in Figure 2.12) accumulates larger total tendency (area under the curve) than a less sticky one (dashed curve in Figure 2.12). The computation is also efficient, $\cos(\theta_i)$ is simply the dot product $N \cdot U_i$. Figure 2.13 shows a dusty sphere with a directional dust source applied from above using the cosine function stated before.

2.5.2 External Factors

To model the influence by external factors like coverage or scraping, the predicted tendency T_S is multiplied by a function α, which tells how much the predicted tendency is realized in the actual environment.

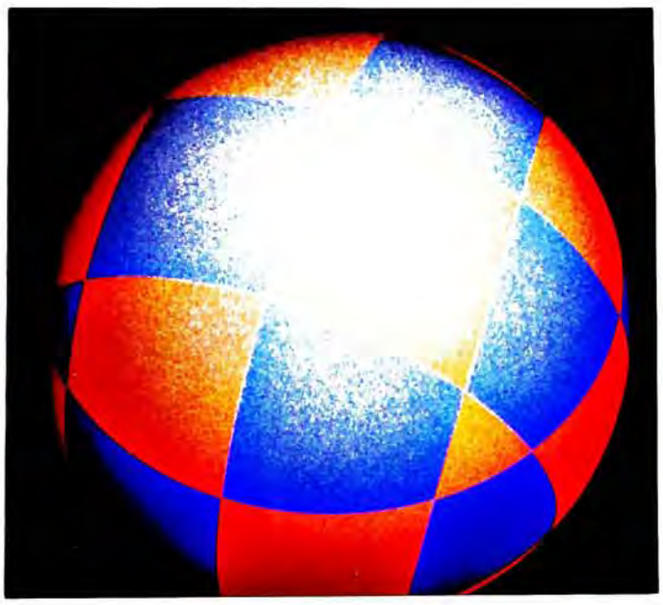

Figure 2.13: A dusty sphere with one directional dust source applied from the top. $\alpha = 1$ and $\eta = 1$.

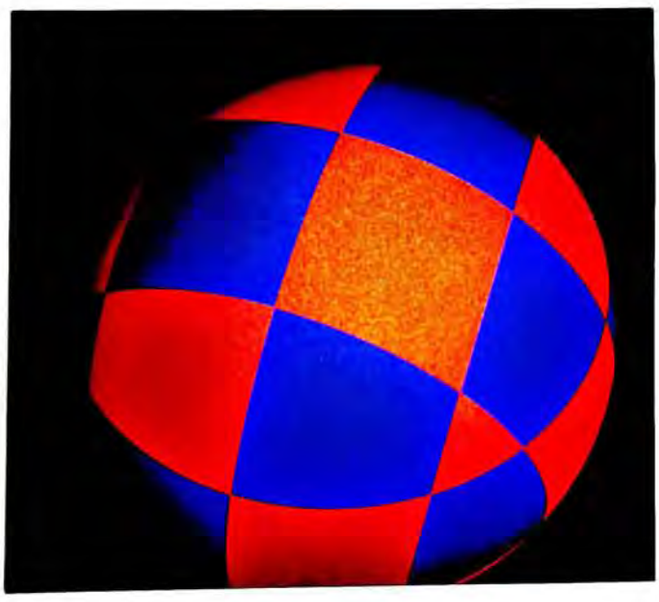

Figure 2.14: The same dusty sphere. $\alpha = 0.6$ and $\eta = 1$.

Figure 2.13 and 2.14 show a dusty sphere generated by multiplying the predicted tendency with different α. The value of α is kept constant throughout the whole surface of the sphere in these two examples. In practice, α varies from point to point.

The value of α is dependent on the geometry of the object and can be further influenced by other factors like coverage and scraping.

Surface Exposure

Consider the object in Figure 2.15. Area B is less exposed to air than area A. As area B has less exposure than area A, the dust amount that area B may *receive* is less. On the other hand, area B may eventually *result* in more dust accumulation - since area B is less exposed, the dust particles accumulated on it have less chance of being removed by wind or scraping. Hence the dust amount of area B is larger than that of area A.

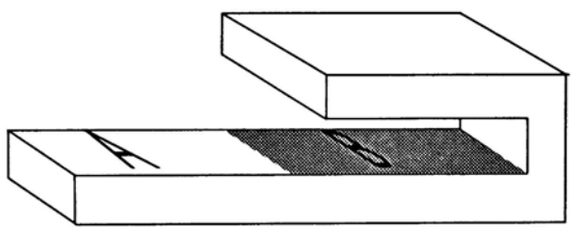

Figure 2.15: Object with a portion in 'shadow'.

In order to model both cases, the term α can be written as a function of surface exposure ξ.

$$\alpha = \begin{cases} 0, & \alpha' < 0 \\ \alpha', & 0 \leq \alpha' \end{cases} \tag{2.7}$$

where $\alpha' = \alpha_o - r_0\xi$ and

α_o is a factor modelling the global external effects, s.t. $\alpha_o \in \Re$.

ξ is the surface exposure, such that $0 \leq \xi \leq 1$. Larger value means more exposed.

r_0 is a scaling factor to scale the effect of surface exposure and $r_0 \in \Re$.

When r_0 is positive, the surface exposure is negatively related to α. This models the case that area A accumulates less dust than area B. Figure 2.16 shows one example with negative surface exposure effect. When r_0 is negative, the surface exposure is positively related to α. This models the opposite case. Figure 2.17 shows the same object with positive exposure effect.

Figure 2.16: The effect of negative surface exposure (i.e. $r_0 > 0$): less-exposed area received more dust.

We have described in section 2.4.1 a method to determine ξ. However, it is expensive to be accurate. We present a more economic approach here. Instead of finding an accurate ξ at each surface point of interest. We can make use of the fuzzy property of dust layer. We only need a very rough approximation of ξ at each surface point by emitting only a few number of rays in random (for the purpose of anti-aliasing) direction in the upper hemisphere of surface point P.

Figure 2.17: The effect of positive surface exposure (i.e. $r_0 < 0$): less-exposed area accumulated less dust.

Then, an inaccurate ξ can be evaluated by equation 2.5 at each surface point of interest. Although ξ is inaccurate in the microscopic level, it is accurate in the macroscopic level (i.e. on average). This can be shown by Figure 2.18 and 2.19, which are generated using 1 and 5 random rays respectively. The value of surface exposure would have a greater variation, if fewer rays are emitted. This would also result in a more scattered appearance.

Scraping

Effects due to scraping are modelled using a dust mapping technique, which is a variant of texture mapping [BLIN76, HECK86]. The surface dust amount is changed according to the pattern in the dust map, which is just a texture map. The equation of α is now:

$$\alpha = \begin{cases} 0, & \alpha' < 0 \\ \alpha', & 0 \leq \alpha' \end{cases} \tag{2.8}$$

Figure 2.18: A dusty door-plate with the word *Dust!* on it. One random ray is casted per pixel in determining the surface exposure.

Figure 2.19: The same door-plate. Five random rays are casted per pixel to determine the surface exposure.

where $\alpha' = \alpha_o - r_0\xi + (b_l + p(b_u - b_l))$ and

p is the image pixel value, s.t. $0 \leq p \leq 1$.

$[b_u, b_l]$ is the interval where p is mapped to. It is used

to control the perturbation effect of p. $b_u, b_l \in \Re$ and $b_l \leq b_u$.

Figure 2.20 shows how one specific scanline in the dust map (left diagram in Figure 2.20) perturbs the term α (right diagram in Figure 2.20). Figure 2.21 shows the dust pattern on a sphere modified with the dust map on the left of Figure 2.20.

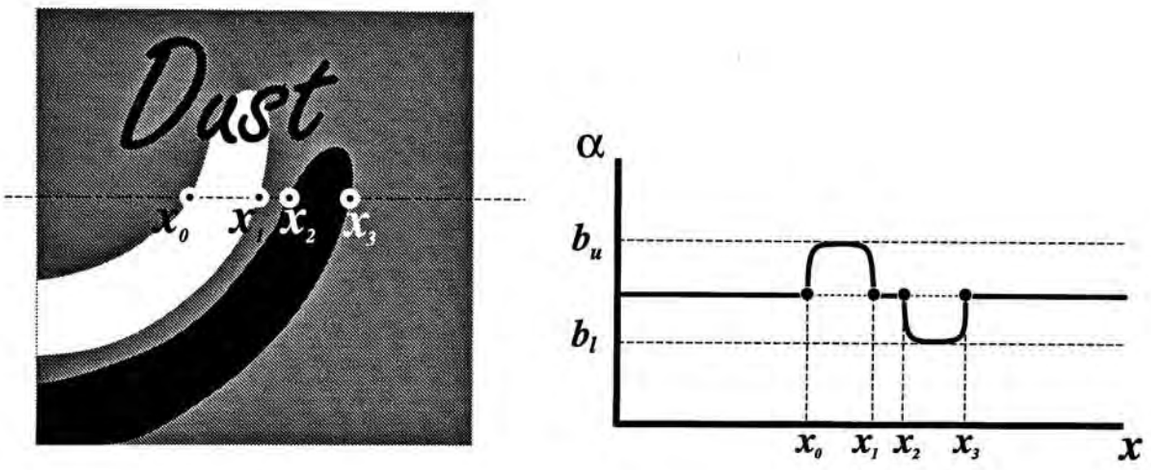

Figure 2.20: The dust map used to perturb the tendency of dust accumulation on the sphere.

Figure 2.21: The dusty sphere modified with a dust map in the previous figure.

2.5.3 Generation of Fuzzy Dust Layer

It is possible to apply the final tendency value T as the thickness parameter to Blinn's light reflection function [BLIN82]. A simpler approach, however, is adopted in our implementation. The final tendency T calculated by equation 2.2 is further perturbed by Perlin's noise function (described in section 2.4.2) to give a fuzzy appearance. The final surface property for rendering are then linearly interpolated with the perturbed value between the original object surface property and the surface property when the surface is fully covered with dust particles (see Equation 2.9). The latter is predefined. This linear interpolation of surface property is just a simplified approximation. All synthetic dusty images (Figure 2.13, 2.14, 2.16, 2.17, 2.18, 2.19, 2.21 and 2.22) are generated using this simplified approach.

$$S_{final} = T'S_{dust} + (1 - T')S_{object} \tag{2.9}$$

where $T' = T \cdot noise(P)$ and

S_{final} is the final surface properties,
S_{dust} is the surface properties when fully covered with dust,
S_{object} is the surface properties when free of dust,
T is the final tendency calculated by Equation 2.2,
$noise()$ is the Perlin's noise function which returns value in $[0, 1]$,
P is the positional vector.

A more accurate approach is to precompute an array of BRDF (Bidirectional Reflectance Distribution Function) tables [KAJI85, WEST92, WARD92]. Different tables record the BRDF of the surface patch covered with different amount of dust particles. On rendering, the actual reflectance can be interpolated among these BRDF tables using the final tendency T.

Figure 2.22: A dusty X-wing fighter 'sprayed' with six point sources.

2.5.4 Implementation Issues

All images are generated using a modified public domain ray tracer, Rayshade ver 4.0.6 Enhanced 2. We have implemented the dust modelling technique as a texture module in Rayshade. Although we implement the technique in a ray tracing based program, the technique can actually be used with other rendering methods, like Z buffer [WATT92] and radiosity [GORA84, COHE85]. In our current implementation, the dust pattern is generated on the fly during rendering. It can, however, be generated in the preprocess phase and stored as a texture map for subsequent use.

2.6 Modelling of Scratching

Scratching is another type of common surface imperfection in real life. It models the appearance of surface when part of the surface layer is peeled off by external forces. Physical modelling seems impratical due to the expensive simulation. We shall present here an empirical technique employing the framework described in section 2.4 to model scratching.

The determination of the tendency of a surface element being peeled off is

similar to that of dust accumulation, except that one more external factor takes place. It is the surface curvature.

Since the calculation of predicted tendency due to a scratch source is similar to that of dust accumulation in Section 2.5.1, we shall concentrate on the discussion of the external factor.

2.6.1 External Factor

The effect of surface exposure and scraping is discussed in Section 2.5.2. We shall not repeat here. In this section, we focus on one factor which is specific to scratching, surface curvature.

Surface Curvature

Paint on a protrusive surface is more likely to be peeled off than that on a flat one. The scratches usually start from the protrusive area and then propagate to the surroundings. This is because the protrusive nature increases the chance of being attacked by external forces. Note that surface exposure alone cannot account for such effect. Consider a convex surface and a flat surface (Figure 2.23). Both of them can be completely exposed, but paint on the convex one has a larger tendency of being peeled off.

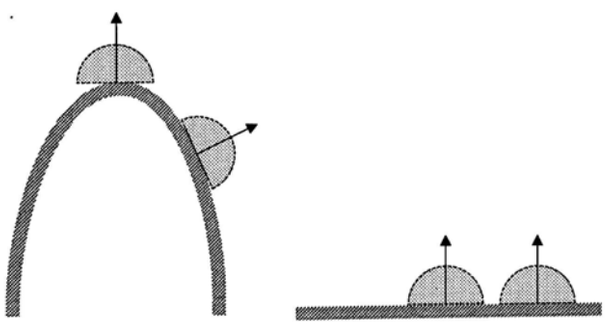

Figure 2.23: Paint on the convex surface has a larger tendency of being peel off than that on a flat one, even though they have the same surface exposure.

The definition of surface curvature is defined as follows. Without loss of generality, we can represent any surface as a parametric surface. For each plane containing the normal at a particular point P on the surface, the curvature κ of the intersection curve between the plane and the surface at point P can be determined. As the plane is rotated about the normal, the curvature changes. It can be showed that there exists distinct directions for which the curvature is a minimum (κ_{min}) and maximum (κ_{max}). They are known as principal curvatures. Two combinations of the principal curvatures are useful, they are the average curvature H and the Gaussian curvature K.

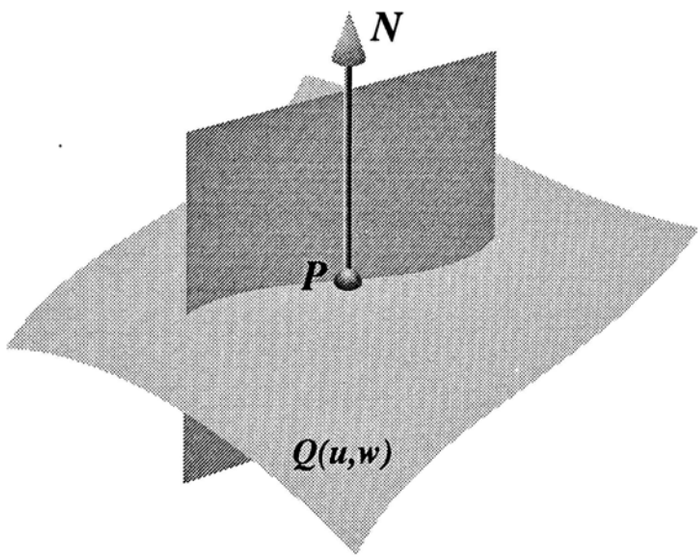

Figure 2.24: The curvature at surface point P is defined as the curvature of the curve of intersection of the plane containing the normal and surface $Q(u, w)$.

$$H = \frac{\kappa_{min} + \kappa_{max}}{2} \tag{2.10}$$

$$K = \kappa_{min}\kappa_{max} \tag{2.11}$$

Dill [DILL81] showed that the average curvature and the Gaussian curvature of a biparametric surface $Q(u, w)$ are,

$$H = \frac{A|Q_w|^2 - 2BQ_u \cdot Q_w + C|Q_u|^2}{2|Q_u \times Q_w|^3} \tag{2.12}$$

$$K = \frac{AC - B^2}{|Q_u \times Q_w|^4} \tag{2.13}$$

where

$$\begin{aligned} A &= (Q_u \times Q_w) \cdot Q_{uu} \\ B &= (Q_u \times Q_w) \cdot Q_{uw} \\ C &= (Q_u \times Q_w) \cdot Q_{ww} \\ Q_u &= \frac{\partial Q}{\partial u} \\ Q_w &= \frac{\partial Q}{\partial w} \\ Q_{uu} &= \frac{\partial^2 Q}{\partial u^2} \\ Q_{ww} &= \frac{\partial^2 Q}{\partial w^2} \\ Q_{uw} &= \frac{\partial^2 Q}{\partial u \partial w} \end{aligned}$$

Figure 2.25 and 2.26 show the average and Gaussian curvature values on a teapot respectively. The red color indicates the surface position is highly curved while the white color indicates the surface is flat.

Figure 2.25: Average curvature on a teapot.

Figure 2.26: Gaussian curvature on the same teapot.

Now we can use either average or Gaussian curvature or both in the calculation of the external factor function α,

$$\alpha = \begin{cases} 0, & \alpha' < 0 \\ \alpha', & 0 \leq \alpha' \end{cases} \tag{2.14}$$

where $\alpha' = \alpha_o - r_0\xi + r_1\kappa_a + r_2\kappa_G + (b_l + p(b_u - b_l))$ and

r_0, r_1 and r_2 are scaling constants $\in \Re$,

κ_a is the average curvature $\in \Re$,

κ_G is the Gaussian curvature $\in \Re$.

2.6.2 Generation of Chaotic Scratch Patterns

Similar to the generation of dust patterns in Section 2.5.3, we use solid texturing to generate scratch patterns. However, we use the fractal fractional Brownian motion (fBm) this time. Instead of interpolation of surface properties, a thresholding approach is used due to the discrete nature of scratches. Whenever the product of T and fBm exceeds a user-defined threshold, the outer surface layer is peeled off. Hence the properties of the inner surface layer is used for shading. This can be generalized to the case of multiple surface layers,

$$S_{final} = \begin{cases} S_{layer_0} & \text{if } 0 \leq T' < \lambda_0, \\ S_{layer_1} & \text{if } \lambda_0 \leq T' < \lambda_1, \\ \cdots & \\ S_{layer_i} & \text{if } \lambda_{i-1} \leq T' < \lambda_i, \\ \cdots & \\ S_{layer_m} & \text{if } \lambda_{m-1} \leq T' < \lambda_m, \end{cases} \tag{2.15}$$

where $T' = T \cdot fBm(P)$ and

$m + 1$ is the total number of of surface layers,

S_{final} is the final surface properties,

S_{layer_i} is the surface properties of the i-th layer below the outermost one S_{layer_0},

λ_i is the user-defined threshold for i-th layer, s.t. $0 \leq \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_m \leq 1$

T is the final tendency calculated by Equation 2.2,

$fBm()$ is the fractional Brownian motion. $fBm()$ returns value in $[0, 1]$,

P is the positional vector.

Figure 2.27 shows a scratched teapot generated by this method. Figure 2.28 shows the same teapot with a point-formed scratch source applied near the bottom. This teapot has only two surface layers.

2.6.3 Implementation Issues

Surface Representation

A specific type of biparametric surface, Bezier surface patches [ROGE90], is used to model object. It is frequently used in computer graphics. It is given by

$$Q(u, w) = \sum_{i=0}^{n_B} \sum_{j=0}^{m_B} B_{i,j} J_{n_B,i}(u) K_{m_B,j}(w) \tag{2.16}$$

Figure 2.27: A scratched teapot. The Gaussian curvature effect is exaggerated by setting the r_2 to a large value and r_1 to zero.

Figure 2.28: The scratched teapot with a point form scratch source applied near the bottom of the teapot.

where

$$J_{n,i}(u) = \begin{pmatrix} n \\ i \end{pmatrix} u^i (1-u)^{n-i}$$

$$K_{m,j}(w) = \begin{pmatrix} m \\ j \end{pmatrix} w^j (1-w)^{m-j}$$

and

n_B, m_B are one less than the number of vertices in the u and w directions

respectively,

$B_{i,j}$'s are the vertices of the $n_B \times m_B$ control points.

In practice, 4×4 control points are used to specify one Bezier surface patch. Bezier surface patch is usually converted to a mesh of polygons before rendering due to the inefficiency of direct rendering of Bezier patch. Recursive subdivision [LANE80] and forward differencing [WALL90] are two common methods to subdivide a Bezier patch into polygons. In our implementation, we use forward differencing to subdivide patches into triangles, since it is more efficient asymptotically.

Curvature Interpolation

In order to speed up the calculation of curvature, we do not calculate the curvature at every surface point. We calculate the average and Gaussian curvatures only at the vertices of each subdivided polygon. The curvature values of the interior points within the polygon are approximated by interpolating among the curvatures at vertices. This is similar to the case of Gouraud shading [GOUR71] which interpolates the intensities and the case of Phong shading [PHON75] which interpolates the surface normals. Although this approximation may not be accurate, it is sufficient for our purpose.

Rendering

We also implement the method in the ray tracer, Rayshade 4.0.6 Enhanced 2, as a texture module. Just like the modelling of dust accumulation, the scratch pattern is generated on the fly during rendering. However, the scratch pattern can be generated in the preprocess phase and stored as a texture map for subsequent use.

Chapter 3

An Improved Space Filling Curve Halftoning Technique

3.1 Introduction

Some classical halftoning techniques, like ordered dither [BAYE73], error diffusion [FLOY76] and dot diffusion [KNUT87] introduce artifacts into the resultant bilevel image. Figure 3.10(b) shows the noticeable regular patterns introduced by ordered dither. Figure 3.10(c) shows the snake-like patterns created by error diffusion. Figure 3.10(d) shows the dot patterns introduced by dot diffusion. These artifacts look not just unpleasant, but also misleading to the viewer. Many newer techniques [WITT82, ULIC87, GEIS90, GEIS93, VELH91] have been proposed to reduce these artifacts.

Another important problem in digital halftoning is the deviation from the ideal intensity due to the smudging of printed dots. This problem is especially important in high resolution printing. While one can correct this by determining the transfer function of a particular device through calibration [GOER87], a better halftoning method could provide a wider range of intensities. Moreover, the accuracy of the transfer function depends on the consistency of the printing

conditions, e.g. quality of the paper used, type of the ink, etc. This method may be inaccurate when, for example, the printer is running out of ink, or paper with different degree of absorbability is used. By clustering the printing dots, we can reduce perimeters of blackened regions which are roughly proportional to the smudging error. Hence it is a more reliable means to ensure the dithered images to have better quality. The traditional ordered dither, Knuth's smooth dot diffusion [KNUT87] and the clustered-dot space filling curve halftoning method proposed by Velho and Gomes [VELH91] all have the clustering capability.

Most digital halftoning techniques can either reduce artifacts or reduce printing smudge error, but not both. The clustered-dot space filling curve halftoning method can, however, reduce both artifacts and printing smudge error. The space filling curve halftoning method is attractive because of its pleasant smooth grains in the resultant image and the aperiodicity of the halftone pattern. The clustering ability of the algorithm, hence the amount of smudging, can also be controlled by a single parameter.

Clustering the dots in the clustered-dot space filling curve method would however excessively blur the image. We propose two improvements, selective precipitation and adaptive clustering, to the clustered-dot space filling curve halftoning method. Results from improved method are compared with the images dithered by clustered-dot space filling curve method and other popular halftoning techniques.

Besides comparing these results subjectively, we also compared the quality of halftone images based on a Gibbs measure [GEIS93] and the total length of perimeters of the blackened areas, which gives a rough measure to the potential amount of smudging. We use them as the objective measures of the halftone quality of various methods.

3.2 Review on Some Halftoning Techniques

On the subject of digital halftoning, much has been written in the past. Ordered dither [BAYE73], error diffusion [FLOY76] and other coloured noise based techniques [GEIS90, ULIC87], dot diffusion [KNUT87] and the more recent space filling curve based techniques [WITT82, VELH91, ZHAN93] are some of the more well-known techniques.

3.2.1 Ordered Dither

Ordered dither [BAYE73] is probably the most commonly used algorithm. The basic algorithm is to tile the whole image with a dither matrix D of threshold values, say

$$D = \begin{bmatrix} 19/32 & 25/32 & 27/32 & 31/32 \\ 21/32 & 5/32 & 3/32 & 17/32 \\ 23/32 & 7/32 & 1/32 & 15/32 \\ 29/32 & 9/32 & 11/32 & 13/32 \end{bmatrix}$$

Whenever the pixel value (range from 0 to 1, 0 is white, 1 is black) in the original image exceeds the corresponding threshold value, a corresponding pixel in the dithered image will be turned on (value 1).

Its popularity is due to the simplicity of the algorithm and its flexibility in being able to produce various halftone screen effects at various angles and densities by simply changing the dither matrix. Another important advantage offered by the method is its ability to cluster the black dots so that the effect of ink smudging can be reduced. Moreover, it can be implemented in parallel. Furthermore, high frequency features in the original image can usually be preserved thus allowing sharp edges to remain sharp in the halftone image. However, the technique tends to smooth out the subtle low frequency details. The halftone resulted also suffers from a correlated periodicity which would produce clear gray level bands on a smooth gray ramp (Figure 3.17(a)).

3.2.2 Error Diffusion and Dither with Blue Noise

Error diffusion [FLOY76] is an elegant scheme designed to preserve the local average gray level in the halftone image. The method distributes the quantization error in a pixel to its neighbourhood so that the quantization error in one pixel would compensate for that arised in another one. Each pixel propagates the error to its right, lower left, below and lower right neighbours in a fixed ratio. Here is the pseudocode,

```
for i = 1 to n
begin
    for j = 1 to n
    begin
        if input[i,j] < 0.5
        begin
            output[i,j] = 0
        end
        else
        begin
            output[i,j] = 1
        end
        error = input[i,j] - output[i,j]
        input[i,j+1] = input[i,j+1] + (error * 7/16)
        input[i+1,j-1] = input[i+1,j-1] + (error * 3/16)
        input[i+1,j] = input[i+1,j] + (error * 5/16)
        input[i+1,j+1] = input[i+1,j+1] + (error * 1/16)
    end
end
```

The variables `input` and `output` are 2D arrays holding the original grayscale image and the dithered image respectively. The pixel value is between 0 and 1. The variable `error` contains the propagated error.

One problem with the original error diffusion method is that the fixed error propagation pattern (or error filter) would produce annoying visual artifacts on flat areas or on smoothly changing gray ramps. By introducing a certain degree of randomness into the error propagation ratio and by alternating the

raster scanning direction on different scanlines, this artifact can generally be eliminated. These modified schemes with randomness are classified as blue noise (high frequency white noise) halftoning methods [ULIC87]. Blue noise methods produce patterns that are aperiodic and radially symmetrical. Because the low frequency components are not present, the resulting image would only appear smooth when viewed from a reasonable distance.

However, the fatal problem with these methods is that they produce dispersed dots which tend to smudge and darken the final image excessively when printed on high resolution devices (Figure 3.16(b) and 3.17(b)).

3.2.3 Dot Diffusion

Knuth [KNUT87] combines ordered dither and error diffusion to develop the dot diffusion method. It has both the clustering ability of ordered dither and the error propagation ability of the error diffusion scheme. The technique has the additional advantage of being able to run in parallel. However, due to the fixed diffusion pattern within a cluster, periodic regular patterns are present in the resultant halftone image (Figure 3.10(d)).

3.2.4 Halftoning Along Space Filling Traversal

A space filling curve traversal is a continuous trace that passes through all pixels in the image exactly once. It was first discovered by Peano in 1890. Some classic space filling curves are the Peano curve (Figure 3.1), the Hilbert curve (Figure 3.2) and the Sierpinski curve. In 1982, Witten and Neal [WITT82] proposed a halftoning algorithm based on space filling curves traversal. The algorithm traverses the whole gray scale image along a space filling curve. The Hilbert curve is particularly suitable for use in halftoning for its higher spatial coherence compared with the other two [VOOR91].

The basic idea of the algorithm is to perform error diffusion along the space

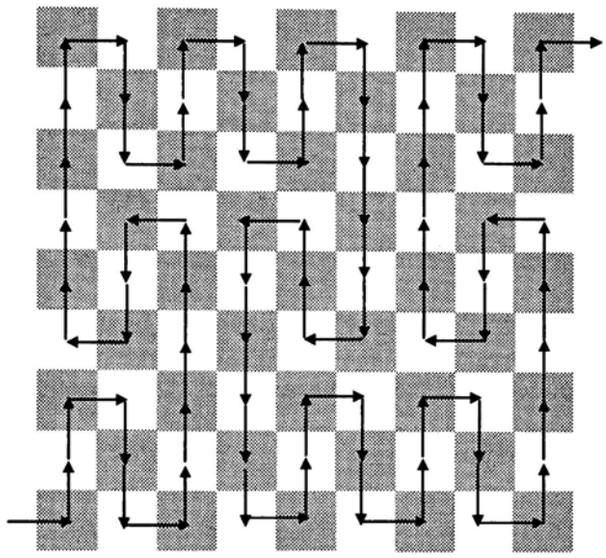

Figure 3.1: Traversing an image along a Peano Curve.

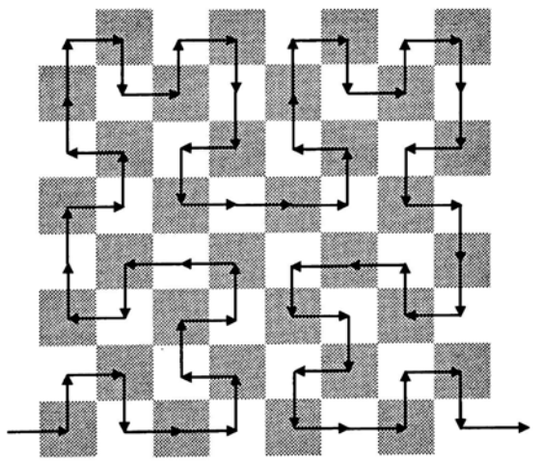

Figure 3.2: Traversing an image along a Hilbert curve.

filling curve. Instead of propagating the quantization error in a pixel to 4 neighbour pixels as in original error diffusion, the error is transferred to the next pixel on the path. Just as error diffusion, the algorithm suffers from the smudging problem due to its dispersed-dot property.

Velho and Gomes proposed a clustered-dot version [VELH91] in 1991. As the program traverses the image, it accumulates pixel values. Whenever a certain fixed number of pixels, which form a *cluster*, have been scanned through, the program generates a number of black dots according to the accumulated gray value in this cluster. According to the original pseudocode in their paper, the black dots are always precipitated at the beginning of each cluster. The error between the actual accumulated gray value and the intensity of generated black pixels is propagated to the next cluster. The pseudocode is,

Given: The size of cluster is `N`.

```
select a space filling curve path
accumulator = 0
while there is pixel to process
begin
      move forward N pixels along the path
      move backward N pixels and accumulate gray value
      move forward N pixels and generate dots:
            if accumulator >= 1
            begin
                  accumulator = accumulator - 1
                  output pixel is on
            end
            else
            begin
                  output pixel is off
            end
end
```

The clustered-dot version uses the parameter *cluster size*, `N`, to control the smudging error. As the cluster size increases, more black dots are connected due to the intertwined nature of space filling curves (Figure 3.1 and 3.2), the total perimeter of blackened area decreases, hence reducing the printing smudge error which depends on the total perimeter. Witten's original algorithm is just a special case when the cluster size is equal to one pixel.

Figure 3.11(a) and 3.11(b) are the same cat image dithered by space filling curve halftoning method with cluster sizes of 1 pixel (i.e. no clustering) and 9 pixels respectively. Both images are free from having periodic regular patterns.

One problem with space filling curve halftoning techniques is that the curve may not fit the dither image. For instances, Hilbert curve traverses image with resolution $2^n \times 2^n$, Peano curve traverses image of $3^n \times 3^n$. Cole [COLE90] generalizes the Peano curve and developed murray curves. Murray curve fits image with resolution $m \times n$, such that m and n are odd and can be factorized. Wyvill and McNaughton [WYVI91] further extend murray curve to develop Geoff curve which can fit image with width and height both greater than seven.

Clustering minimizes the image darkening problem due to smudging of printed dots and dot gain. However, as the cluster size increases, the dithered image becomes blurrier and small details are lost (Figure 3.11(b), 3.12(b), 3.13(b) and 3.14(b)).

3.2.5 Space Diffusion

Space diffusion [ZHAN93] is a recently proposed scheme based on Knuth's idea of dot diffusion but using a space filling curve pattern to perform the dot diffusion. The method is identical to Knuth's but with a different diffusion matrix. The method has the same advantages of being parallel and that no regular periodic pattern would appear on the resultant images. However, the technique as described in the paper no longer have the clustering capability of the clustered-dot

space filling curve method and hence is unsuitable for high resolution printing. If a clustering scheme similar to the clustered-dot space filling curve version is used, the method would suffer from the same excessive blurring problem.

3.3 Improvements on the Clustered-Dot Space Filling Halftoning Method

One disadvantage of the clustered-dot space filling curve halftoning method is that images dithered by this method are usually blurrier than those dithered by other halftone techniques like ordered dither and error diffusion. One can improve the quality of dither image by preprocessing the original grayscale image [ULIC87]. Such preprocessing techniques include sharpening, smoothing, increasing contrast and gamma correction. However, all these methods change the gray value of each pixel in the original grayscale image, hence producing unfaithful resultant images. Preprocessing the grayscale image does not solve the underlying problems in the clustered-dot space filling halftoning method. We recognize that the blurring is due to the poor precipitation scheme of black pixels and the fixation of cluster size in the original method. Aiming at these two causes, we are proposing two improvements on the clustered-dot space filling curve halftoning method, namely, *selective precipitation* and *adaptive clustering.*

3.3.1 Selective Precipitation

The clustered-dot space filling curve halftoning algorithm (Section 3.2.4) precipitates the black pixels at a fixed location, say, at the beginning of each cluster. This results in a poor approximation (Figure 3.3(b)) to the original image when the original gray values (Figure 3.3(a)) in a particular cluster are not gathered around that fixed location. Although Velho and Gomes had briefly suggested in their paper that the white subregion can be centred at the pixel with the

highest intensity in order to preserve details, this may still result in a poor approximation (Figure 3.4(b)).

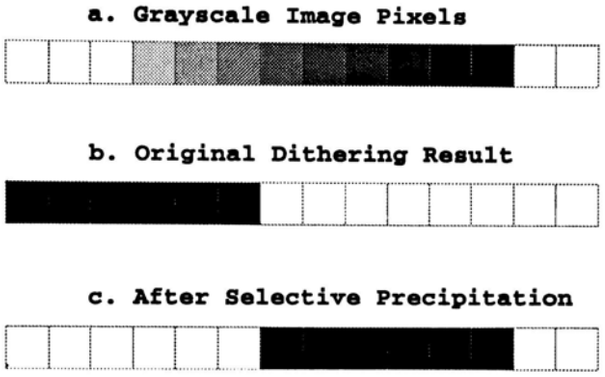

Figure 3.3: (a) A straightened cluster along a space filling curve in the original gray scale image. (b) Resulting halftone based on the clustered-dot space filling curve halftoning method if we precipitate black pixels at the beginning of the cluster. (c) A better approximation.

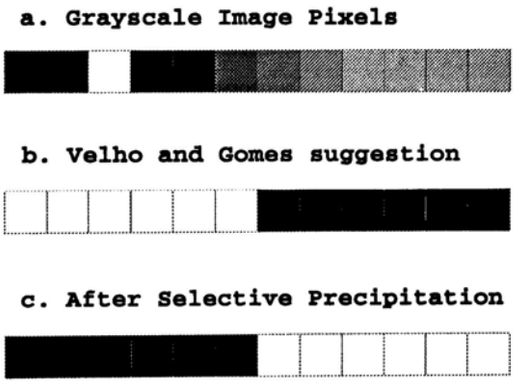

Figure 3.4: (a) A straightened cluster along a space filling curve in the original grayscale image. (b) Result based on Velho and Gomes's suggestion. (c) Result using selective precipitation.

We propose a method of selectively placing the output black pixels over the area with the highest total gray value. This would give a better approximation to the original gray value distribution. We call this technique *selective precipitation.* The algorithm is as follows. The number of black pixels to be output in the current cluster is determined in the same way as the clustered-dot space filling

curve method. This number is then used as the length of a moving window which shifts within the halftone cluster. The gray values of the grayscale image pixels within the moving window are summed and recorded with the position of the window. Our objective is to find the position in the cluster such that the sum of gray values of ⌊graysum⌋ consecutive pixels is the highest. That position is where we start to precipitate the black dots. The algorithm is sketched below:

Assumptions:

1. *Assume the pixel has gray value between 0 and 1.*
2. *In order to simplify the formulation, we number each image pixel in the order of the space filling curve traversal. We can reference any pixel on the original image using the notation* `input[i]` *just like an one dimension array.*

Input:

1. `clusterstart` *is the index number of the first element of the current cluster.*
2. `graysum` *is the sum of gray value inside the current cluster.*
3. `clustersize` *is the size of the cluster.*

Output:

1. *The quantization error is returned through the variable* `graysum`.

Algorithm:

```
winlen = ⌊graysum⌋
graysum = graysum - winlen
winsum = maxsum = 0
winstart = clusterstart
for i = winstart to (winstart+winlen-1)
begin
     winsum = winsum + input[i]
end
while (winstart+winlen) - clusterstart < clustersize
begin
     if maxsum < winsum
     begin
```

```
            maxsum = winsum
            rightplace = winstart
        end
        winsum = winsum - input[winstart] + input[winstart+winlen]
        winstart = winstart + 1
    end
```

Output `winlen` *number of black pixels starting from* `rightplace`

The variable `input` is an array holding the gray values of the original image pixels. After running the above process, we can output black dots starting from `rightplace`. The time complexity of this process is linear since each pixel inside the cluster is accessed at most twice.

The improvement over the clustered-dot space filling curve method is quite substantial. Figure 3.11(c) shows the result of halftoning the cat image using the improved space filling curve method with selective precipitation (cluster size is 9 pixels). Comparing to Figure 3.11(b) which uses the clustered-dot space filling curve method (cluster size is also 9 pixels), our image seems sharper and more subtle details are perceivable. Notice that patterns on the cat's cheeks can be now seen clearly which are blurred in Figure 3.11(b). Similar improvements can be observed in Figure 3.12(c), 3.13(c) and 3.14(c). All of them seem sharper and preserve more details which are blurred in the images dithered by the clustered-dot space filling curve method. Another effect resulted from the selective precipitation technique is that the effective cluster size is usually higher. This is because when a cluster boundary overlaps a darker area in the original image, the resultant black dots in clusters on both sides of the cluster boundary would stick to the boundary. In the clustered-dot space filling method, this could never happen however.

3.3.2 Adaptive Clustering

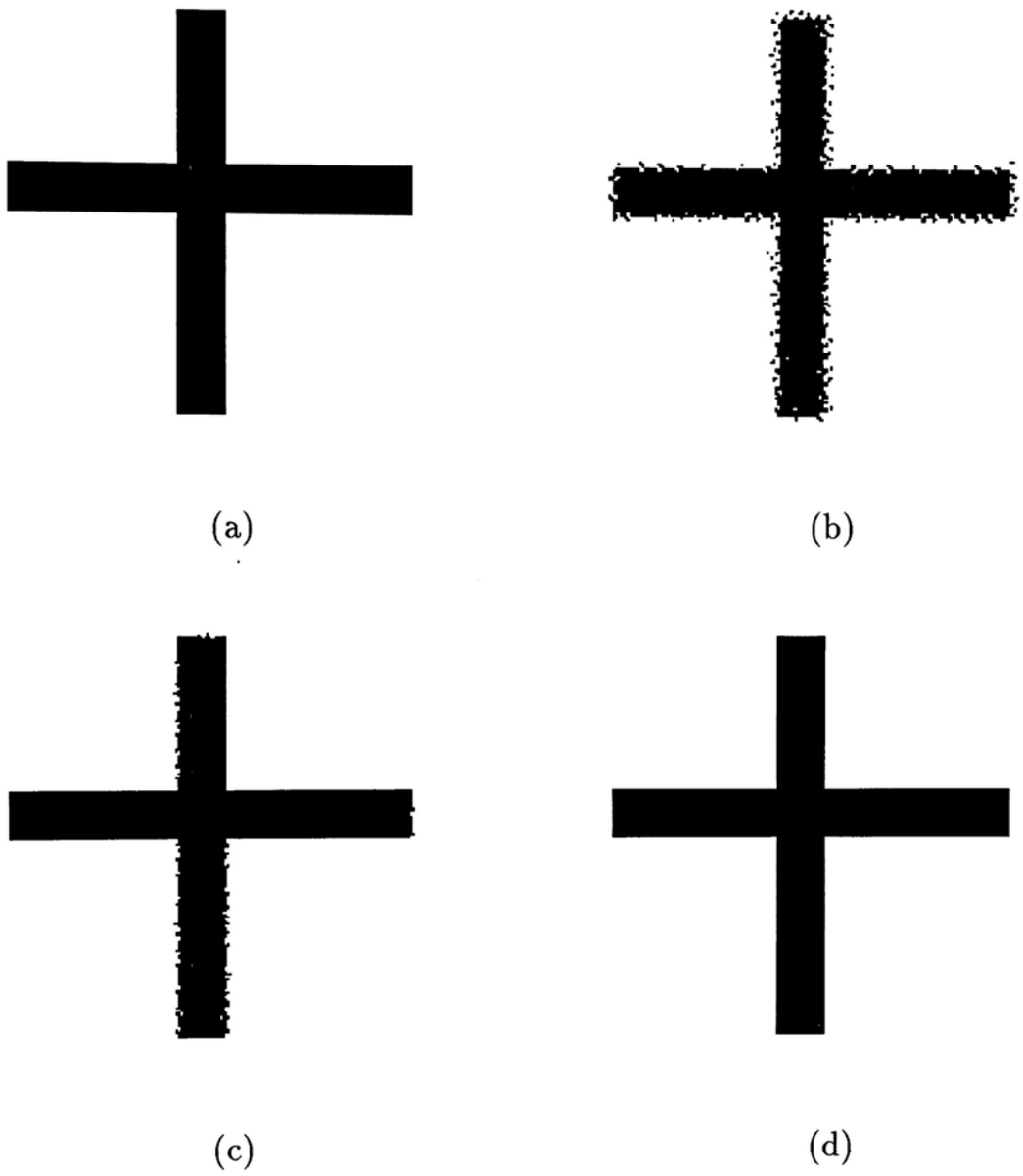

Figure 3.5: (a) Gray scale image of a cross. (b) Dithered by the space filling curve with cluster size = 9 pixels. (c) With selective precipitation. (d) After adaptive clustering with threshold T = 0.012.

Even with selective precipitation, we still cannot ensure that the sharpness of high frequency edges are preserved. Figure 3.5(a) shows an original grayscale image of a cross with very sharp edges. If we dither this image using the clustered-dot space filling curve halftoning method with 9-pixel clusters, we

get Figure 3.5(b), which is blurred excessively. The blurring is reduced (Figure 3.5(c)) on using selective precipitation. However, one can still see that some edges are still a little bit fuzzy. This example shows that rigid positioning of black pixels is not the only cause of the blurring.

Another factor which causes the blurring is the rigid grouping of output black pixels. Figure 3.6 shows the case. The original gray values are accumulated on both ends of the cluster. In such a case, the only thing selective precipitation can do is to generate black pixels on the end with higher total gray value. If we divide the cluster into 2 smaller subclusters however and perform the selective precipitation process in both subclusters, a better approximation can be resulted.

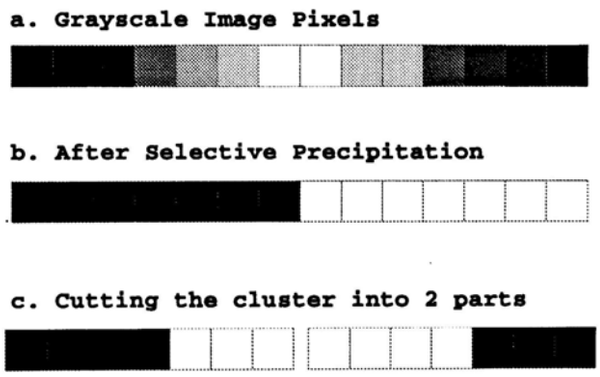

Figure 3.6: (a) The straightened cluster along the space filling curve in the original gray scale image. (b) The result after running the selective precipitation. (c) better result if the cluster is divided into 2 smaller clusters.

Of course one can decrease the size of all clusters to reduce the occurrence of the cases as in Figure 3.6(a). However, a smaller cluster size also means a weaker capability of reducing the effect of printer smudge. This is a dilemma. To make a trade-off, we use smaller clusters only when necessary. That is, instead of using a fixed cluster size as in the clustered-dot space filling curve method, we allow the cluster size to vary.

One way to decide where to subdivide the cluster is to detect whether the

case like Figure 3.6(a) occurs. Instead, we decide to subdivide a cluster wherever a sharp edge is detected. Since human eyes are more sensitive to high frequency changes, blurring phenomena on sharp edges are more noticeable (Figure 3.5(c)). This partitioning of clusters at sharp edges would be able to preserve sharp details. We call this subdivision technique *adaptive clustering.*

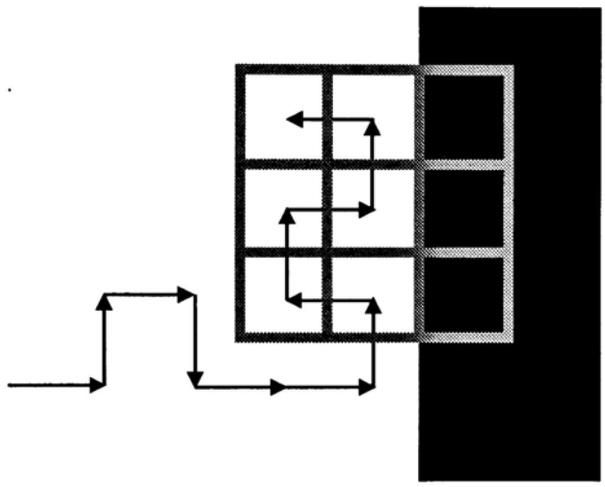

Figure 3.7: 2D edge detector may detect edge which is not intercepted by the space filling curve.

Even though a 2D edge detection filter can be applied to the image to identify edges, it is computationally expensive. Furthermore, edges which would not be intercepted by the space filling curve (e.g. tangential to the space filling curve or at the cluster boundary, see Figure 3.7) would also be detected which are unnecessary. Since the space filling curve visits all pixels in the image exactly once, it effectively scales down the 2D image into a 1D chain of signal. It is good enough to employ just a 1D filter along the space filling curve. We find that detecting edges using 1D filter is good enough for visual purpose. It may be due to the intertwined behaviour of space filling curve. This argument can be shown in all example images (Figure 3.11(d), 3.12(d), 3.13(d), 3.14(d), 3.15(c) and 3.15(d)) in this thesis. The standard 1D Laplacian of the Gaussian filter [JAIN89] (Figure 3.8) is applied to this chain of signals to recognize sharp edges.

This filter is the negation of the second derivative of the Gaussian function with respect to the input signal x:

$$\frac{e^{\frac{-x^2}{2\sigma^2}}}{\sigma^3\sqrt{2\pi}}(1-\frac{x^2}{\sigma^2})$$

where σ is the standard deviation.

In our implementation, the filter kernel has a width of 7 pixels. In order to convolve the filter on previous and next three pixels (total 7 pixels), we set σ to 1.

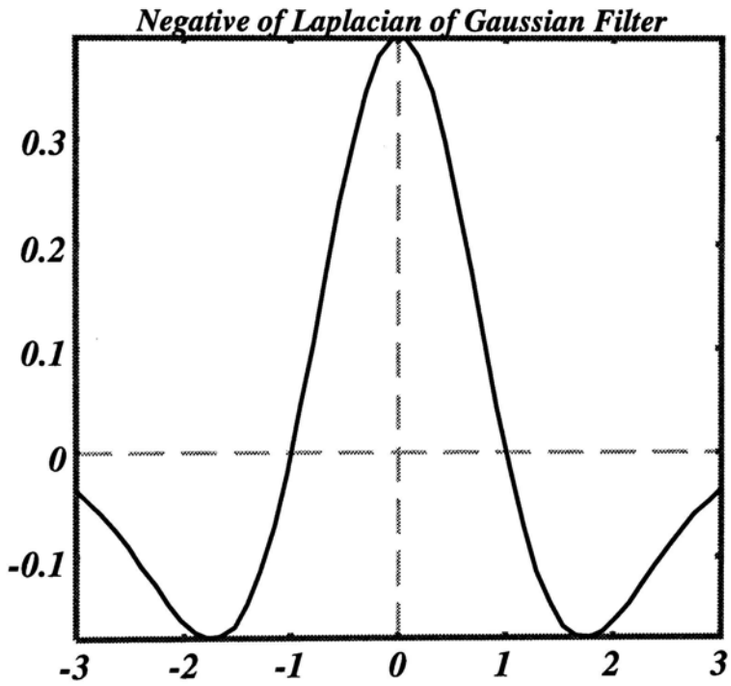

Figure 3.8: 1D Negative of the Laplacian of the Gaussian filter.

The algorithm to perform adaptive clustering is as follows. The image pixels are scanned through in the original space filling order. A cluster is formed whenever `N` (the maximum cluster size) pixels have been traversed or a sharp edge is encountered, whichever comes earlier. Selective precipitation is then performed within the cluster. The pseudocode is shown below:

Given:

1. *maximum cluster size* `N`,
2. *threshold* `T`.

Algorithm:

```
select a space filling curve pattern
graysum = 0
clustersize = 0
current = clusterstart = 0
while current < no. of pixel
begin
    currconv = result of convoluting the negative of
               the Laplacian of the Gaussian filter
               to the image input[] with
               input[current] (current pixel)
               as the centre.
    graysum = graysum + input[current]
    clustersize = clustersize + 1
    if abs(currconv-lastconv)>T or clustersize>N
    begin
        perform selective precipitation with parameters
        graysum, clustersize and clusterstart.
        clustersize = 0
        clusterstart = current
    end
    lastconv = currconv
    current = current + 1
end
```

Figure 3.5(d) shows the cross dithered by this improved filling curve method (with adaptive clustering and selective precipitation). The maximum cluster size `N` is set to 9 pixels. Our new algorithm completely eliminates all blurring of sharp edges in this example. There are more samples produced by our algorithm with `N`=9 and `T`=0.012, they are Figure 3.11(d), 3.12(d), 3.13(d) and 3.14(d). Notice those small details in the background of F16 factory image (Figure 3.13(d)) can be seen quite clearly. All of them retain sharp edges that would have been blurred

using selective precipitation alone. They look sharper than the corresponding images produced by the clustered-dot space filling curve halftoning method. Moreover, the algorithm would not break clusters into subclusters when the image contains only gradual intensity change, hence maximizing the clustering capability.

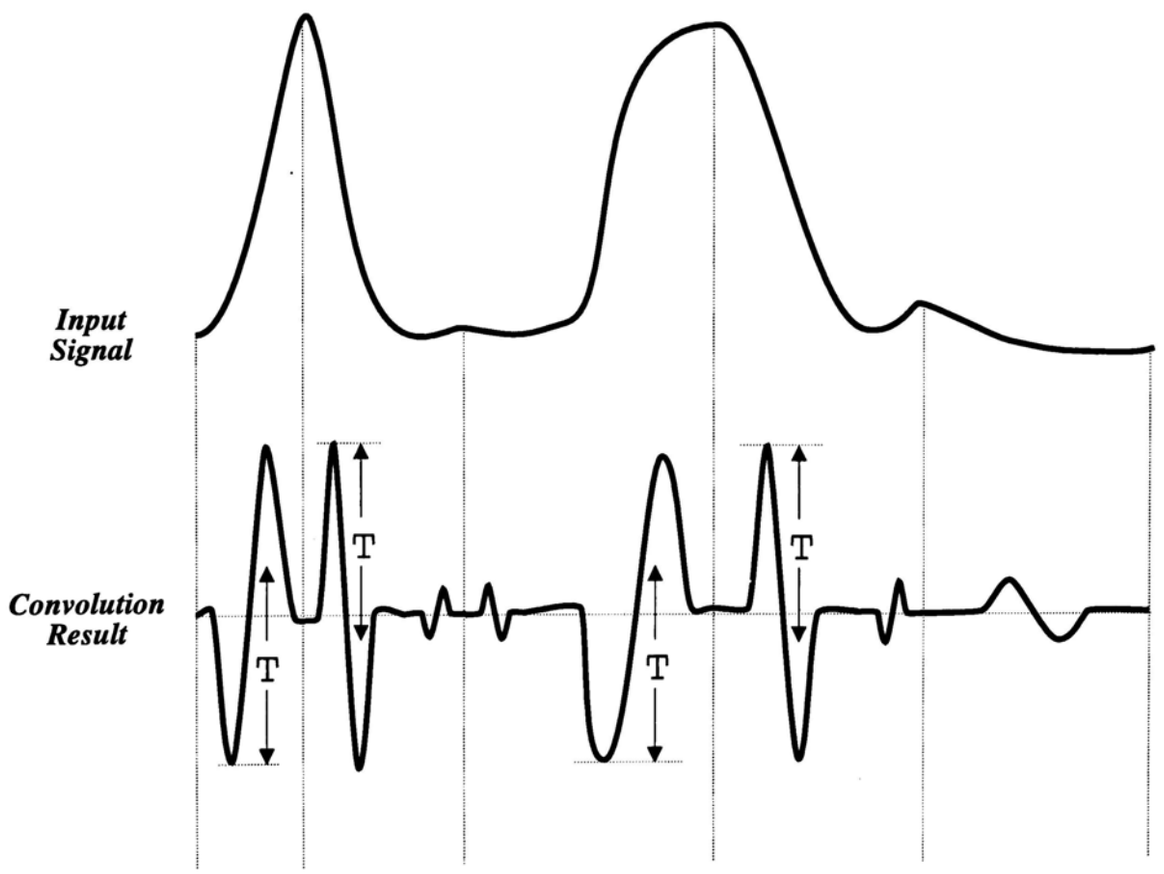

Figure 3.9: The upper curve is the 1D input gray scale image signal. The lower curve is the corresponding convolution result after applying the 1D negative of the Laplacian of Gaussian filter.

The sensitivity of the edge detection filter also affects the resulting halftone image. We control this with a user-defined threshold T (Figure 3.9). This value can also be determined automatically using the technique presented in [SCHL91]. A lower threshold would allow more edges to be detected and results in potentially smaller cluster sizes. The effect is demonstrated in Figure 3.15.

Figure 3.15(a) is the teapot image dithered by the clustered-dot space filling curve method with a 55-pixel cluster. Figure 3.15(b), 3.15(c) and 3.15(d) are images dithered by our improved algorithm using T=100, T=0.08 and T=0.012 respectively. All of them use a maximum cluster size N of 55 pixels. The smaller the T is, the more edges are detected. Hence more small details are visible.

3.4 Comparison With Other Methods

The resultant halftone images created using various methods are shown in Figure 3.11–3.17. The quality improvement resulted from our enhanced methods is promising.

3.4.1 Low Resolution Observations

Figure 3.10 and Figure 3.11 show dithered images of a cat using various halftoning methods. Images produced by ordered dither (Figure 3.10)(b), original error diffusion (Figure 3.10(c)) and dot diffusion (Figure 3.10(d)) all produced some forms of artifacts. Ordered dither and dot diffusion generate periodic regular patterns, while error diffusion produces snake-like patterns. On the other hand, images dithered by space filling curve halftoning methods reduce artifacts. Figure 3.11(a) is dithered using dispersed-dot space filling curve halftoning method (cluster size = 1 pixel), while Figure 3.11(b) is dithered by clustered-dot space filling curve halftoning method with cluster size = 9 pixels. One can notice that the image dithered by large cluster is excessively blurred. Notice that the tongue, the ear, the detail on the forehead and the whisker in Figure 3.11(b) cannot be seen clearly. Images dithered by both selective precipitation and adaptive clustering preserve more details than those dithered with selective precipitation alone. This argument is verified by Figure 3.13(d), where the original grayscale contains many small details.

3.4.2 High Resolution Printing Results

Figure 3.16 and 3.17 show results of printing on a 600 dpi printer. The halftone images are created respectively using ordered dither (order 4×4), error diffusion, the clustered-dot space filling curve method (N=15) and our improved version with both adaptive clustering and selective precipitation (N=15, T=0.012). The quality of our images is clearly superior than the clustered-dot space filling curve method and more subtle details that are smoothed out in the ordered dither version are also visible.

The excessive darkening of images by error diffusion is apparent from these outputs. The artifact of using ordered dither to distribute the errors is also demonstrated by the appearance of clear bands (false contours) on the gray scale ramp in Figure 3.17(a).

3.4.3 Analytical Comparison

Gibbs Model

Geist proposes the use of a maximum-entropy Gibbs measure as an objective indicator of the quality of dithered images. The mathematical details can be found in [GEIS93]. We do not go into the detailed derivations, but present only how the measure is calculated.

For each pixel, we calculate two terms. The first term represents the correlation between the individual dithered pixel and the underlying grayscale intensity. It is formulated as $(1 - 2w_i)(2V_i - 1)$, where V_i is the gray value of the pixel in the original grayscale image and w_i is the dithered pixel with value 0 or 1. The second term represents the correlation of the neighbourhood of the dithered pixel and the neighbourhood of the corresponding grayscale pixel. This term is formulated as $f(i,j)(1-2w_i)(1-2w_j)$ where i is the current dithered pixel and j is its neighbour. Geist suggested that pixels inside a circle of radius 5 pixels of

the current pixel are considered as neighbours. The *computational energy* $E(\theta)$ of a dithered image is the sum of above two terms of all pixels. Hence it is,

$$E(\theta) = -\sum_{i=0}^{n-1} \theta_i I_i - \frac{1}{2}(\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} T_{ij}\theta_i\theta_j)$$

where

$$T_{ij} = T_{ji} = \begin{cases} -f(i,j) & \text{if } i, j \text{ are neighbors,} \\ 0 & \text{otherwise.} \end{cases}$$

$$I_i = (2V_i - 1) \tag{3.1}$$

$$\theta_i = (2w_i - 1) \tag{3.2}$$

We have measured results generated by our improved algorithm using the Gibbs measure. The values $E(\theta)$ of all dithered images are shown in Table 3.1. Notice that the more negative $E(\theta)$ is, the better the quality of the dithered image is. Among all halftoning algorithm, error diffusion has the lowest energy. However, it does not have clustering ability. The worst is the ordered dither, even though it supports clustering.

As the cluster size increases, $E(\theta)$ increases, hence the quality decreases. In each of four samples, we get a lower $E(\theta)$ when we perform selective precipitation. The energy is further lowered when adaptive clustering is also performed. Among the various space filling curve based algorithms, our improved method gives the best results. Moreover, our method performs even better than the ordered dither, which has clustering ability.

Table 3.2 shows the computational energy when using different threshold T. As the threshold decreases, the filter is more sensitive to edges and more edges are preserved, hence the image quality increases.

Total Perimeter of Black Regions

Another quantity which we have adopted as a measure of the susceptibility to smudging is the total perimeter of black regions in the dithered image. The

	Ordered dithered (4 × 4 kernel)	Error diffusion	Dispersed-dot space filling curve halftoning method (N=1)	Clustered-dot space filling curve halftoning method (N=9)	Improved using selective precipitation (N=9)	Improved using selective precipitation and adaptive clustering with T=0.012
Cat (Fig. 3.11(a))	-24661	-27206	-26150	-24035	-25398	-26292
F14 (Fig. 3.12(a))	-17164	-21000	-19391	-17145	-18118	-18872
F16 factory (Fig. 3.13(a))	-22904	-28095	-24978	-16712	-22606	-26472
Teapot (Fig. 3.14(a))	-17763	-21712	-20178	-17278	-18717	-19143

Table 3.1: Computational energy $E(\theta)$ of different samples dithered by different algorithms.

	Original space filling curve halftoning method with N=55.	Improved with selective precipitation and adaptive clustering with N=55, T=100.	Improved with selective precipitation and adaptive clustering with N=55, T=0.08.	Improved with selective precipitation and adaptive clustering with N=55, T=0.012.
Teapot (Fig. 3.14(a))	-13317	-16056	-16448	-17513

Table 3.2: Computational energy $E(\theta)$ of dithered images dithered with different threshold T.

total perimeter is the number of pixel edges between one black and one white pixel. The smudging error should be roughly proportional to this total perimeter (The geometry of the perimeter would also affect the amount of smudging. We have not taken into account of this factor for simplicity). The smaller the total perimeter is, the weaker the effect of smudging of printed dots is. Table 3.3 shows total perimeters of images dithered by various halftoning methods. Among all samples, error diffusion produces images with the longest total perimeter. This agrees with the fact that it suffers from severe smudging (Figure 3.16(b) and 3.17(b)). The best one (smallest total perimeter) is produced by the space filling curve method with selective precipitation alone. This is due to the increased effective cluster size as explained in Section 3.3.1. Images dithered by space filling curve halftoning method with selective precipitation and adaptive clustering have longer total perimeter, since some clusters are subdivided. However, these values are still comparable to values from ordered dither and the clustered-dot space filling curve method. This shows that adaptive clustering can generally improve the image quality without sacrificing the clustering advantage.

	Error diffusion	Ordered dither (4×4 kernel)	Original space filling curve halftoning method (N=9)	Improved with selective precipitation only (N=9)	Improved with selective precipitation and adaptive clustering with N=9, T=0.012.
Cat	31052	20780	21594	17700	22344
F14	46918	40398	38975	33958	38124
F16 factory	45157	35826	38795	32913	41239
Teapot	48494	31963	34259	29501	31652

Table 3.3: Total perimeter of different samples dithered by different algorithms.

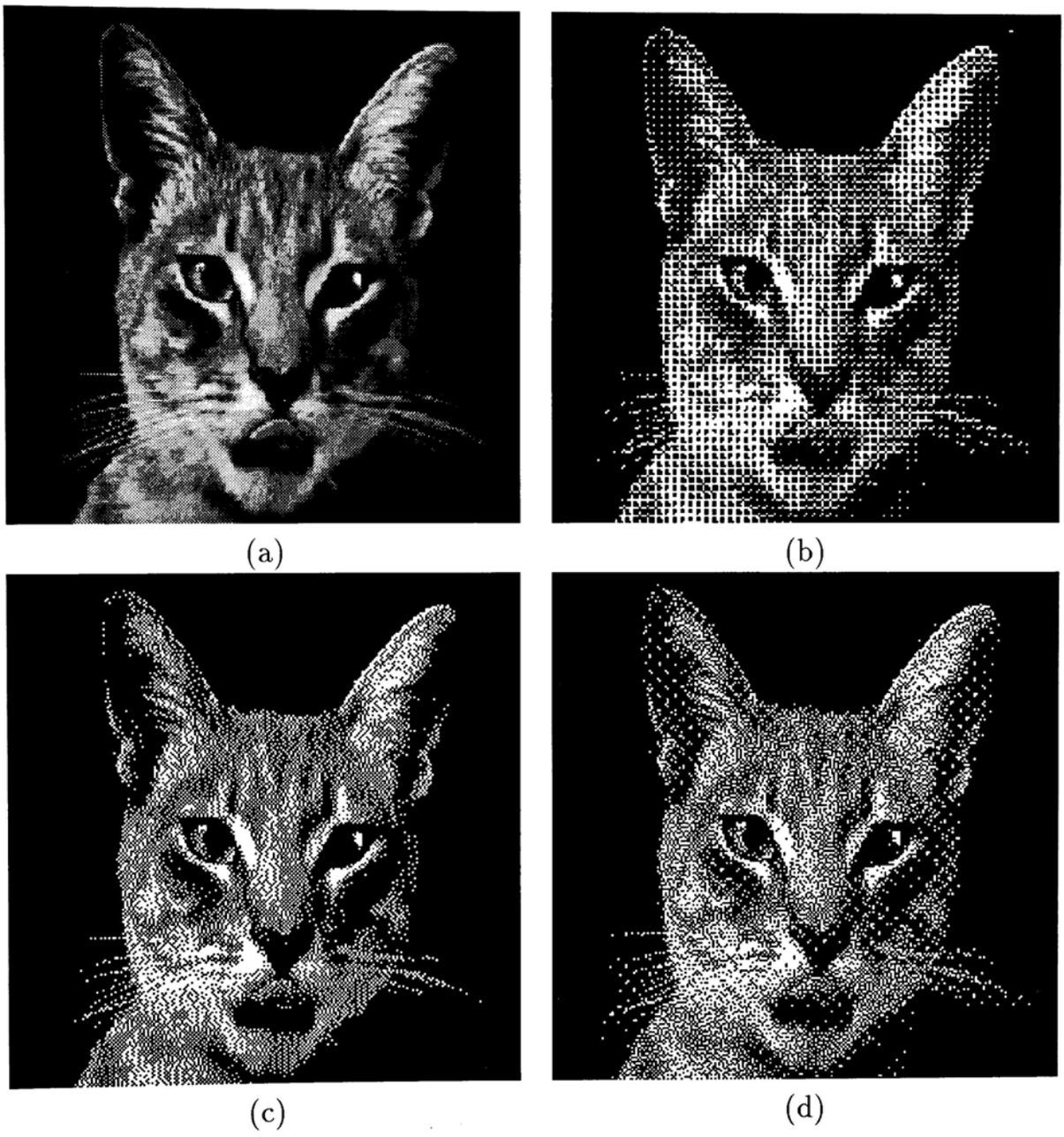

Figure 3.10: (a) Original 256×256 gray scale cat. (b) 256×256 Ordered dithered image applying matrix D. (c) Cat image dithered by error diffusion. (d) Cat image dithered by dot diffusion.

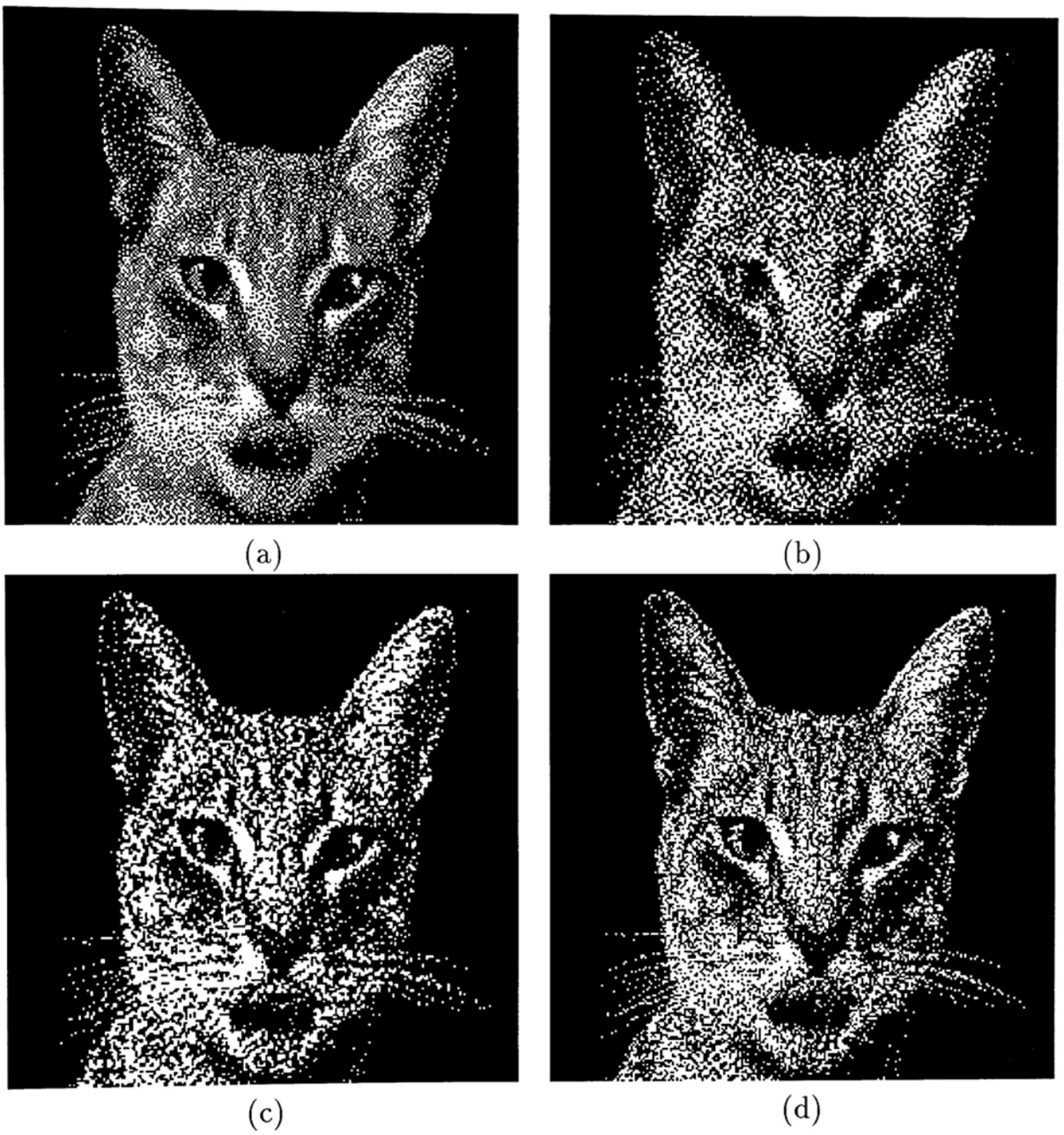

Figure 3.11: (a) Dithered by space filling curve halftoning method with no clustering (1-pixel cluster). (b) Dithered by space filling curve halftoning method with a cluster size `N` of 9 pixels. (c) Dithered by the same process as previous figure, except this image go through an additional selective precipitation process. (d) Dithered by space filling curve halftoning method plus selective precipitation and adaptive clustering with `N`=9, `T`=0.012.

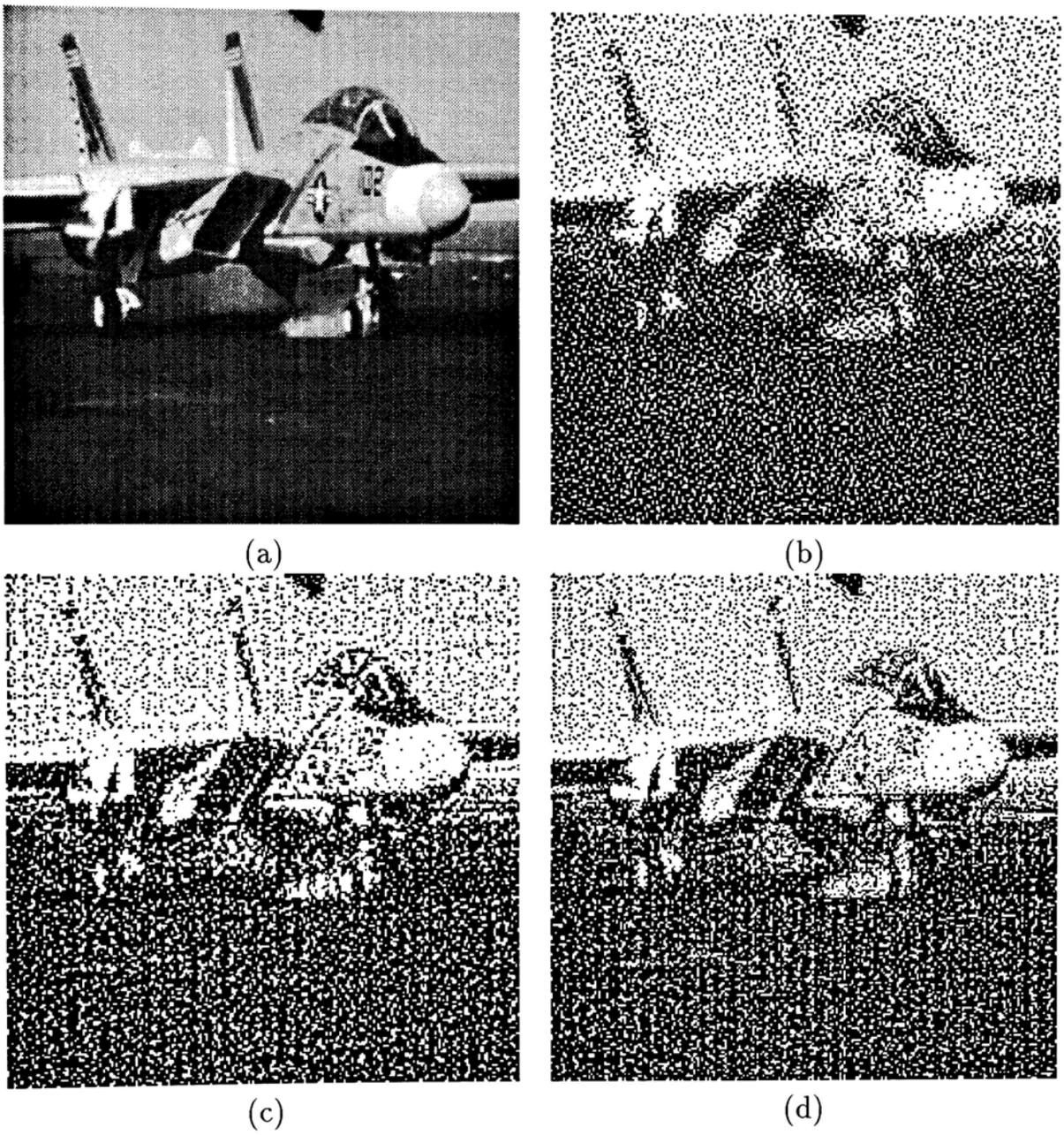

Figure 3.12: (a) 256×256 gray scale image of F14. (b) Dithered by the clustered-dot space filling curve halftoning method. The bands on the ground have totally lost. (c) After selective precipitation process. (d) After selective precipitation and adaptive clustering with N=9, T=0.012. Notice that the vertical bands on the ground are clearly visible.

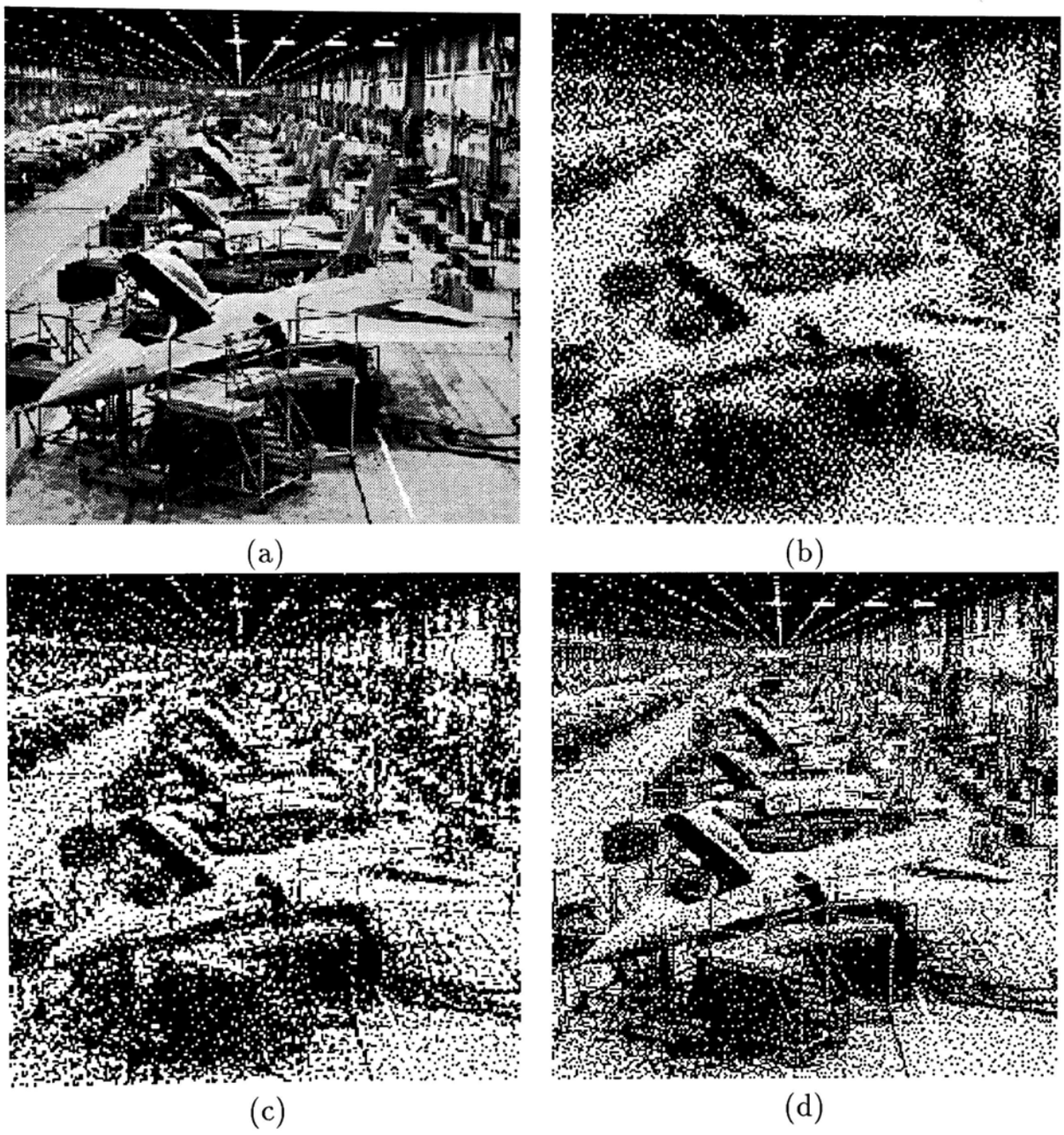

Figure 3.13: (a) 256 × 256 gray scale image of F16 factory. (b) Dithered by the clustered-dot space filling curve halftoning method. (c) After selective precipitation. (d) After selective precipitation and adaptive clustering with N=9, T=0.012.

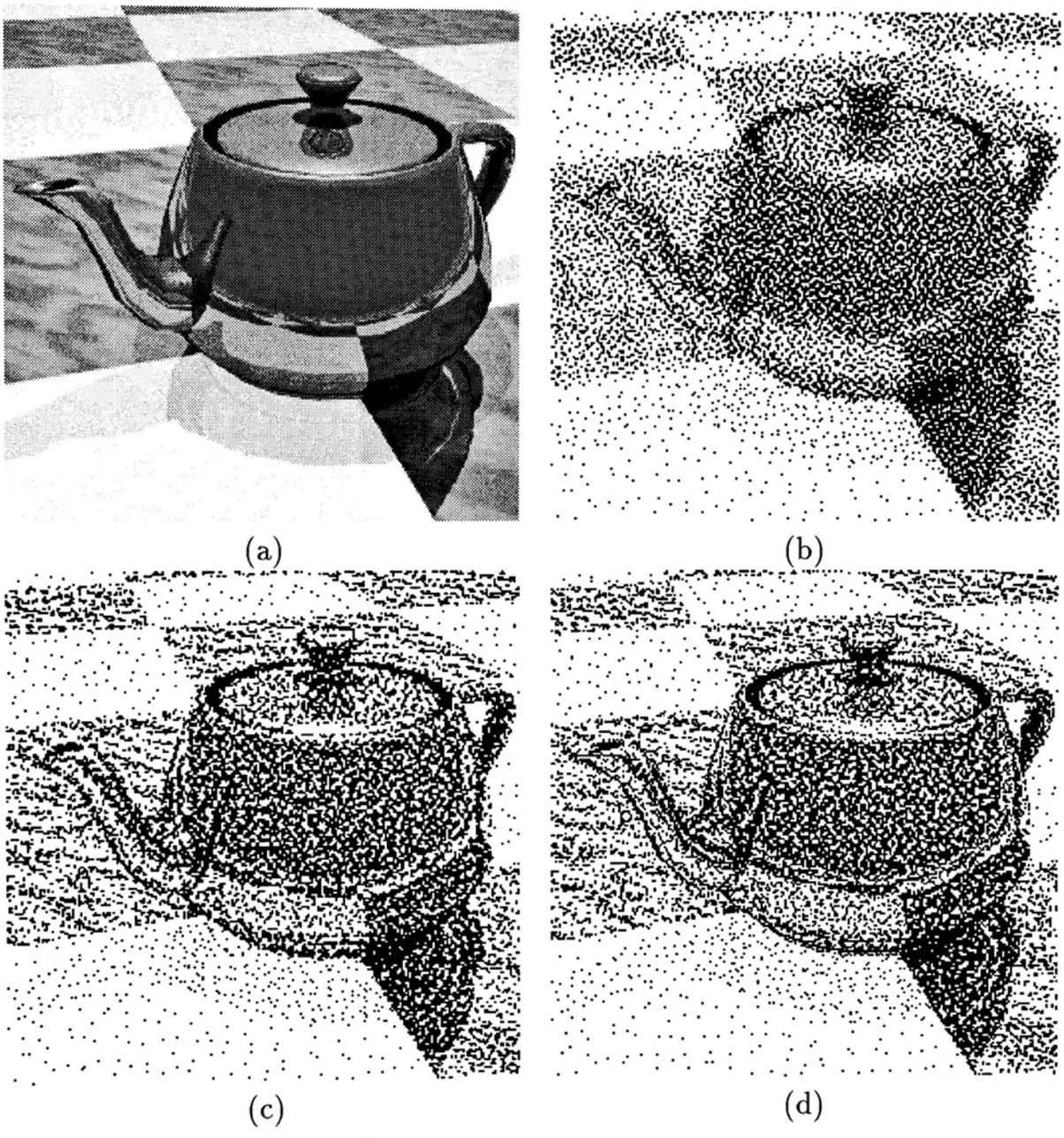

Figure 3.14: (a) 256 × 256 gray scale image of teapot. (b) Dithered by the clustered-dot space filling curve halftoning method. (c) After selective precipitation. (d) After selective precipitation and adaptive clustering with N=9, T=0.012.

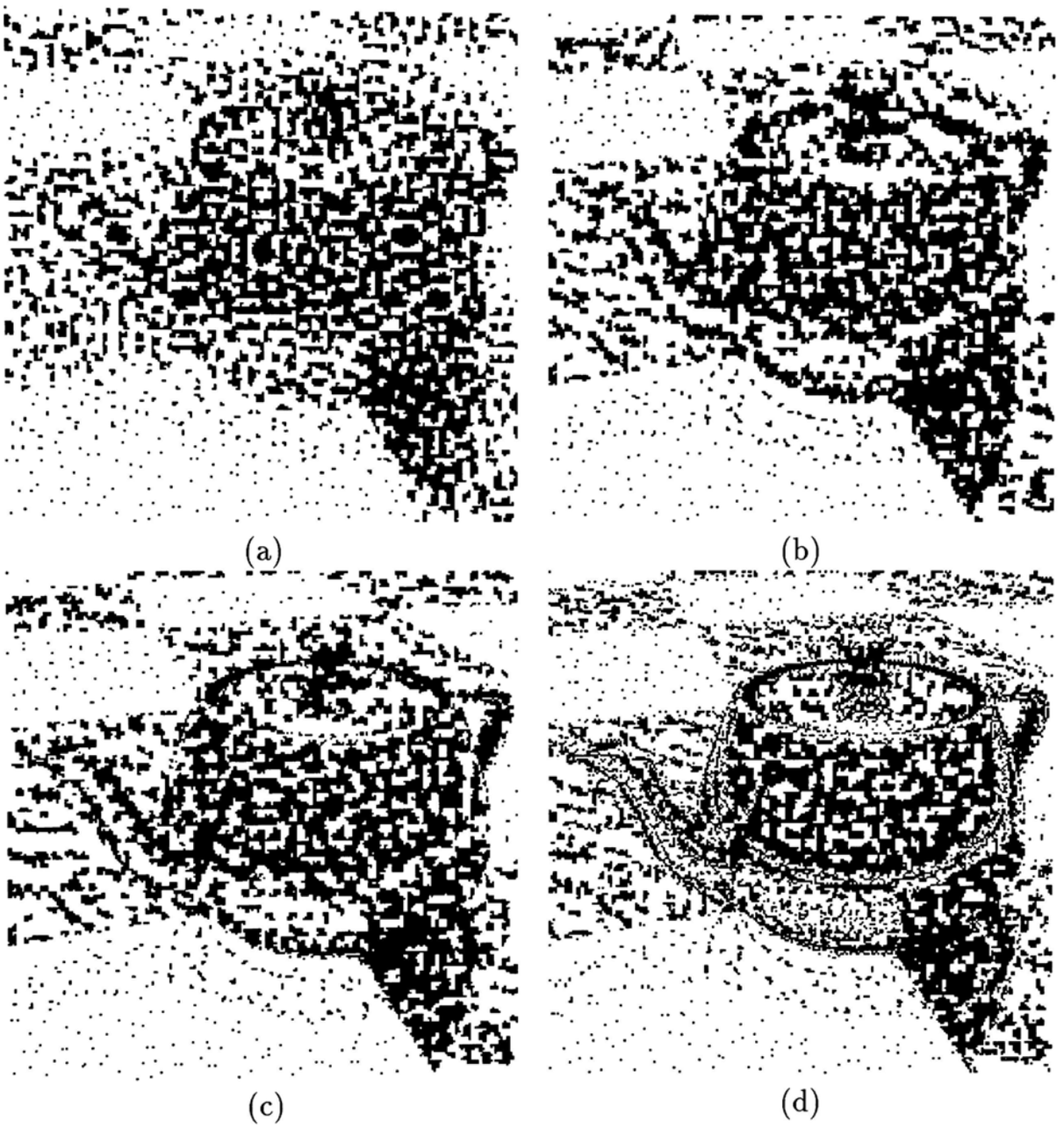

Figure 3.15: (a) Dithered by clustered-dot space filling curve halftoning method with cluster size=55. (b) Dithered by improved space filling curve halftoning method with N=55, T=100. That is detect no edge at all, only selective precipitation. (c) Dithered by improved space filling curve halftoning method with N=55, T=0.08. (d) Dithered by improved space filling curve halftoning method with N=55,T=0.012.

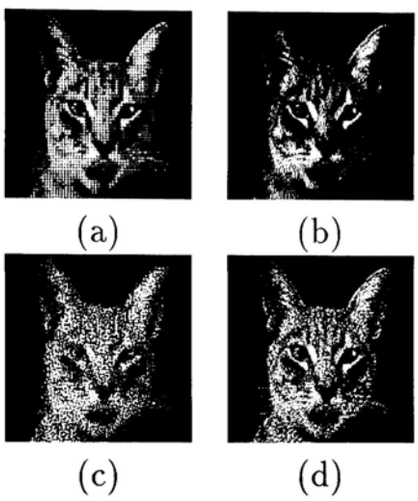

Figure 3.16: The cat image printed on a 600 dpi printer using different halftoning methods. (a) Ordered dither. (b) Error diffusion. (c) Original space filling curve (N=15) (d) Selective precipitation and adaptive clustering (N=15, T=0.012)

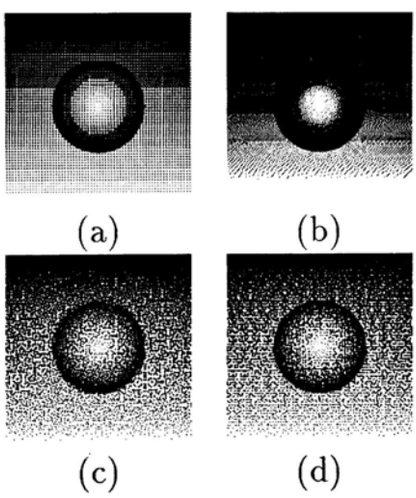

Figure 3.17: A test image with gradually changing gray values printed on a 600 dpi printer using different halftoning methods. (a) Ordered dither. (b) Error diffusion. (c) Original space filling curve (N=15) (d) Selective precipitation and adaptive clustering (N=15, T=0.012)

Chapter 4

Conclusion and Future Work

In this chapter, we draw some conclusions on the two topics discussed previously. Future works are also stated.

4.1 The Modelling of Natural Imperfections

The use of traditional texture mapping to create the appearance of blemished surfaces requires a lot of manual adjustment and is difficult to apply to irregularly shaped objects. The possible distortions during the mapping process further complicate the composition of the desired texture. Furthermore, the distribution of blemishes on different parts of an object cannot be automatically deduced.

The new framework proposed in this thesis requires only a few parameters to control the overall effect of blemishes. The overall appearance can be controlled by placing a few imperfection sources around the object. Small details are automatically generated according to geometrical informations. Hence the user is freed from detail adjustments. The separation of the tendency value calculation from the generation of the final blemish pattern provides the freedom to switch pattern generation methods. Solid texturing is used to generate

blemished patterns due to its simplicity and efficiency.

We have applied the framework to the modelling of dust accumulation and scratching. Some convincing images are resulted. Moreover, the proposed tendency function can be efficiently computed. This can be seen from our implementations: we have implemented the model as a texture module of a ray tracer. Although the tendency value is calculated during rendering, the calculations do not slow down the rendering process significantly. If the generated patterns are stored as texture maps for subsequent use, the rendering process can be speed up further.

As the number of abstract imperfection sources increases, the calculation of tendency value of each surface point may increase. This may not cause serious problem because each object usually has a only few private (or local) imperfection sources associates with it. The imperfection sources of one object cannot affect another object. This is quite different from the case of light source. All light sources defined should be global to all objects in the environment.

In the future, other imperfections, like mould and rust, can be implemented using the proposed framework. A program that allows placing the imperfection sources interactively and storing the resultant pattern as texture map is worth implementing.

To generate more realistic dust patterns, an anisotropic[1] reflection model [KAJI85, CABR87, WARD92, WEST92] seems promising. A natural phenomenon which is quite similar to dust accumulation is snow accumulation. However, snow has significant volume. The next step we want to take would be to model precipitations with volume.

[1]The term *anisotropic* is opposite to the term *isotropic*. An anisotropic surface has different surface properties (e.g. reflection) when it is viewed from different direction.

4.2 An Improved Space Filling Curve Halftoning Technique

We have proposed two methods, selective precipitation and adaptive clustering, to improve the clustered-dot space filling curve halftoning method. The images obtained from our improved method appear sharper than that from the clustered-dot space filling curve method. The sharpness is achieved without any form of pre-filtering. The qualities of the resultant images are also compared analytically using a Gibbs energy measure and total perimeter of black regions in the dithered image. These two measures demonstrate that our improved method does result in better quality halftones without a corresponding increase in susceptibility to printing smudge.

The improved techniques can be applied to the serial clustered-dot space filling curve method as well as the parallel version (space diffusion) proposed by Zhang [ZHAN93].

Our method still inherits some disadvantages of the space filling curve based halftoning method. The method requires more memory to store the whole image due to the irregular traversal of space filling curve. This can be worked around if we store pixels of the image in the order of the space filling curve traversal. This requires additional conversion from traditional image storing format. Moreover, a special display method is also required to display the converted images.

Clustering black pixels for output reduces the occurrence of smudging of printed dots. This is a rather passive approach. It has a disadvantage that different portion of the image may have different intensity error, since the amount of smudging is different. A more aggressive approach is to compensate the intensity error due to smudging. We will do more research along this direction in the future.

Bibliography

[AMAN84] AMANATIDES, J. Ray tracing with cones. In CHRISTIANSEN, H., editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pp. 129–135, July 1984.

[BAYE73] BAYER, B. E. An optimum method for two level renditionof continuous-tone pictures. In *Proceedings of the IEEE International Conference on Communication*, pp. 26–11–26–15, 1973.

[BECK90] BECKET, W. AND BADLER, N. I. Imperfection for realistic image synthesis. *Journal of Visualization and Computer Animation*, 1(1):26–32, Aug. 1990.

[BLIN76] BLINN, J. F. AND NEWELL, M. E. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–546, October 1976.

[BLIN78] BLINN, J. F. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12, pp. 286–292, August 1978.

[BLIN82] BLINN, J. F. Light reflection functions for simulation of clouds and dusty surfaces. In *Computer Graphics (SIGGRAPH '82 Proceedings)*, volume 16, pp. 21–29, July 1982.

[CABR87] CABRAL, B., MAX, N., AND SPRINGMEYER, R. Bidirectional reflection functions from surface bump maps. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pp. 273–281, July 1987.

[CATM74] CATMULL, E. E. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD thesis, Dept. of CS, U. of Utah, Dec. 1974.

[COHE85] COHEN, M. A radiosity method for the realistic image synthesis of complex diffuse environments. Master's thesis, Program of Computer Graphics, Cornell Univ., Aug. 1985.

[COLE90] COLE, A. J. Naive halftoning. In *Proceedings of Computer Graphics International '90*, pp. 203–222, 1990.

[COOK84] COOK, R. L., PORTER, T., AND CARPENTER, L. Distributed ray tracing. In CHRISTIANSEN, H., editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pp. 137–145, July 1984.

[DILL81] DILL, J. C. An application of color graphics to the display of surface curvature. In *Computer Graphics (SIGGRAPH '81 Proceedings)*, volume 15, pp. 153–161, Aug. 1981.

[FLOY76] FLOYD, R. AND STEINBERG, L. An adaptive algorithm for spatial grey scale. In *Proceedings of SID 17*, pp. 75–77, 1976.

[FOLE90] FOLEY, J. D., VAN DAM, A., FEINER, S. K., AND HUGHES, J. F. *Computer Graphics: Principles and Practices (2nd Edition).* Addison Wesley, 1990.

[FOUR82] FOURNIER, A., FUSSELL, D., AND CARPENTER, L. Computer rendering of stochastic models. In *Communications of the ACM*, volume 25, pp. 371–384, June 1982.

[GARD84] GARDNER, G. Y. Simulation of natural scenes using textured quadric surfaces. In CHRISTIANSEN, H., editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pp. 11–20, July 1984.

[GARD85] GARDNER, G. Y. Visual simulation of clouds. In BARSKY, B. A., editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pp. 297–303, July 1985.

[GEIS90] GEIST, R. AND REYNOLDS, R. Colored noise inversion in digital halftoning. In *Proceedings of Graphics Interface '90*, pp. 31–38, May 1990.

[GEIS93] GEIST, R., REYNOLDS, R., AND SUGGS, D. A markovian framework for digital halftoning. *ACM Transactions on Graphics*, 12(2):136–159, April 1993.

[GOER87] GOERTZEL, G. AND THOMPSON, G. R. Digital halftoning on the ibm 4250 printer. *IBM Journal Res. Develop.*, 31(1):2–15, January 1987.

[GORA84] GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. Modeling the interaction of light between diffuse surfaces. In CHRISTIANSEN, H., editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pp. 213–222, July 1984.

[GOUR71] GOURAUD, H. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, June 1971.

[GREE86] GREENE, N. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, Nov. 1986.

[HANR90] HANRAHAN, P. AND HAEBERLI, P. E. Direct WYSIWYG painting and texturing on 3D shapes. In BASKETT, F., editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pp. 215–223, Aug. 1990.

[HECK86] HECKBERT, P. S. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[JAIN89] JAIN, A. K. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[KAJI85] KAJIYA, J. T. Anisotropic reflection models. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pp. 15–21, July 1985.

[KALR91] KALRA, P. K. Fractals and their applications. In THALMANN, M. AND THALMANN, D., editors, *New Trends in Animation and Visualization*, chapter 14, pp. 197–205. John Wiley & Sons, England, 1991.

[KNUT87] KNUTH, D. E. Digital halftones by dot diffusion. *ACM Transactions on Graphics*, 6(4):245–273, 1987.

[LANE80] LANE, J. AND RIESENFELD, R. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Trans. Pattern Analysis Machine Intell.*, 2(1):35–46, 1980.

[MAND77] MANDELBROT, B. B. *The Fractal Geometry of Nature.* W.H. Freeman and Co., New York, rev 1977.

[MAND82] MANDELBROT, B. B. Comment on computer rendering of fractal stochastic models. In *Communications of the ACM*, volume 25, pp. 581–584, August 1982.

[MAX86] MAX, N. L. Atmospheric illumination and shadows. In EVANS, D. C. AND ATHAY, R. J., editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pp. 117–124, Aug. 1986.

[MILL84] MILLER, G. S. AND HOFFMAN, C. R. Illumination and reflection maps: Simulated objects in simulated and real environments. In *SIGGRAPH '84 Advanced Computer Graphics Animation seminar notes.* July 1984.

[NISH87] NISHITA, T., MIYAWAKI, Y., AND NAKAMAE, E. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In STONE, M. C., editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pp. 303–310, July 1987.

[OPPE86] OPPENHEIMER, P. E. Real time design and animation of fractal plants and trees. In EVANS, D. C. AND ATHAY, R. J., editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pp. 55–64, August 1986.

[PEAC85] PEACHEY, D. R. Solid texturing of complex surfaces. In BARSKY, B. A., editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pp. 279–286, July 1985.

[PEIT88] PEITGEN, H.-O. AND SAUPE, D. *The Science of Fractal Images.* Springer-Verlag, 1988.

[PERL85] PERLIN, K. An image synthesizer. In BARSKY, B. A., editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pp. 287–296, July 1985.

[PHON75] PHONG, B.-T. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[ROBE93] ROBERTSON, B. Dinosaur magic. *Computer Graphics World*, pp. 44–52, September 1993.

[ROGE90] ROGERS, D. F. AND ADAMS, J. A. *Mathematical Elements for Computer Graphics.* McGraw-Hill, 1990.

[RUSH87] RUSHMEIER, H. E. AND TORRANCE, K. E. The zonal method for calculating light intensities in the presence of a participating medium. In STONE, M. C., editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pp. 293–302, July 1987.

[SAUP88] SAUPE, D. Algorithms for random fractals. In PEITGEN, H.-O. AND SAUPE, D., editors, *The Science of Fractal Images*, pp. 71–136. Springer-Verlag, 1988.

[SCHL91] SCHLAG, J. Noise thresholding in edge images. In *Graphics Gems II*, pp. 105. Academic Press, 1991.

[SHIN87] SHINYA, M., TAKAHASHI, T., AND NAITO, S. Principles and applications of pencil tracing. In STONE, M. C., editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pp. 45–54, July 1987.

[SHIN89] SHINYA, M., SAITO, T., AND TAKAHASHI, T. Rendering techniques for transparent objects. In *Proceedings of Graphics Interface '89*, pp. 173–82, Toronto, Ontario, June 1989. Canadian Information Processing Society.

[TO77] TO, Y. K. *Basic Principles in Biology.* Hung Fung Book Co. Ltd., 1977.

[ULIC87] ULICHNEY, R. *Digital Halftoning.* MIT Press, 1987.

[UPST89] UPSTILL, S. *The Renderman Companion: A Programmer's Guide to Realistic Computer Graphics.* Addison Wesley, 1989.

[VELH91] VELHO, L. AND GOMES, J. D. Digital halftoning with space filling curves. In SEDERBERG, T. W., editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pp. 81–90, July 1991.

[VOOR91] VOORHIES, D. Space-filling curves and a measure of coherence. In *Graphics Gems II*, pp. 26–30. Academic Press, 1991.

[VOSS85] VOSS, R. F. Random fractal forgeries. In EARNSHAW, R. A., editor, *Fundamental Algorithms for Computer Graphics*, pp. 805–835. Springer-Verlag, 1985.

[VOSS88] VOSS, R. F. Fractals in nature: From characterization to simulation. In PEITGEN, H.-O. AND SAUPE, D., editors, *The Science of Fractal Images*, pp. 22–70. Springer-Verlag, 1988.

[WALL90] WALLIS, B. Tutorial on forward differencing. In *Graphics Gems*, pp. 594–603. Academic Press, 1990.

[WARD92] WARD, G. J. Measuring and modeling anisotropic reflection. In CATMULL, E. E., editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pp. 265–272, July 1992.

[WATT92] WATT, A. AND WATT, M. *Advanced Animation and Rendering Techniques: Theory and Practice.* Addison-Wesley Publishing Company, 1992.

[WEST92] WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. Predicting reflectance functions from complex surfaces. In CATMULL, E. E., editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pp. 255–264, July 1992.

[WITT82] WITTEN, I. H. AND NEAL, R. M. Using peano curves for bilevel display of continuous-tone images. *IEEE Computer Graphics and Applications*, 2:47–52, May 1982.

[WYVI91] WYVILL, G. AND MCNAUGHTON, C. Three plus five makes eight: A simplified approach to haltoning. In PATRIKALAKIS, N. M., editor, *Scientific Visualization of Physical Phenomena (Proceedings of CG International '91)*, pp. 379–392. Springer-Verlag, 1991.

[ZHAN93] ZHANG, Y. AND WEBBER, R. E. Space diffusion: An improved parallel halftoning technique using space-filling curves. In KAJIYA, J. T., editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pp. 305–312, Aug. 1993.