國立交通大學

多媒體工程研究所

碩 士 論 文

基 於 空 間 填 充 曲 線 的 圖 像 藝 術 風 格

An Artistic Rendition of Images Based on Space-filling Curves

研 究 生：廖雅瑩

指導教授：施仁忠　教授

張勤振　教授

中 華 民 國 一 百 零 五 年 八 月

基於空間填充曲線的圖像藝術風格

An Artistic Rendition of Images Based on Space-filling Curves

研 究 生：廖雅瑩　　Student：Ya-Ying Liao

指導教授：施仁忠　　Advisor：Prof. Zen-Chung Shih

張勤振　　Prof. Chin-Chen Chang

國 立 交 通 大 學

多 媒 體 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

August 2016

Hsinchu, Taiwan, Republic of China

中華民國一百零五年八月

國 立 交 通 大 學

博碩士論文紙本論文暨全文電子檔著作權授權書

（提供授權人裝訂於紙本論文書名頁之次頁用）

本授權書所授權之學位論文，為本人於國立交通大學多媒體工程研究所 _____組， 105 學年度第 1 學期取得碩士學位之論文。

論文題目：基於空間填充曲線的圖像藝術風格
指導教授：施仁忠, 張勤振

一、紙本論文著作權授權

■同意立即公開

本人依著作權法第15條第2項第3款之規定(採推定原則即預設圖書館得公開上架閱覽)，同意將本著作，以非專屬、無償授權國立交通大學與國家圖書館，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學圖書館與國家圖書館得以紙本收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行閱覽或列印。

□不同意立即公開

□1. 本論文為本人向經濟部智慧局申請專利的附件之一，請將論文延至____年____月____日公開，申請文號 為： _____。

□2. 本論文已投稿期刊並待審核中，請將論文延至____年____月____日公開。

說明：配合教育部函(100年7月1日臺高(二)字第1000108377號)，紙本不公開期限以5年為限

二、論文全文電子檔著作權授權

本人同意將本著作，以非專屬、無償授權國立交通大學、台灣聯合大學系統圖書館與國家圖書館：基於推動讀者間「資源共享、互惠合作」 之理念，與回饋社會與學術研究之目的，得不限地域、時間與次數，以紙本、光碟或數位化等各種方法收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間：	
中英文摘要(不公開以5年為限)	■ 立即公開
本校及台灣聯合大學系統區域網路	■ 中華民國 106 年 8 月 22 日公開
校外網際網路及國家圖書館	■ 中華民國 106 年 8 月 22 日公開

研究生：龐雅瑩. _____（親筆簽名）

指導教授：施仁忠 _____（親筆簽名）

中華民國 105 年 8 月 29 日

國立交通大學

研究所碩士班

論文口試委員會審定書

本校 多媒體工程 研究所 廖雅瑩 君

所提論文:

基於空間填充曲線的圖像藝術風格

An Artistic Rendition of Images Based on Space-filling Curves

合於碩士資格水準、業經本委員會評審認可。

口試委員：魏德樂 蔡倩延

祝仁忠 張勤振

指導教授：祝仁忠 張勤振

所　　長：彭文志

中 華 民 國 一百零五 年 八 月 一 日

Institute of Multimedia Engineering
College of Computer Science
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

As members of the Final Examination Committee, we certify that we have read the thesis prepared by Ya-Ying Liao entitled An Artistic Rendition of Images Based on Space-filling Curves and recommend that it be accepted as fulfilling the thesis requirement for the Degree of Master of Science.

Committee Members:

Der-Lor Way　　Yu-Ting Tsai

Zen-Chung Shih　　Chin-Chen Chang

Thesis Advisor: Zen-Chung Shih　Chin-Chen Chang

Director: Wen-Hsiao Peng

Date: 2016/8/1

基於空間填充曲線的圖像藝術風格

研究生: 廖雅瑩　　指導教授: 施仁忠 教授

張勤振 教授

國立交通大學多媒體工程研究所

摘　　要

空間填充曲線(Space-filling curve)曾被使用於藝術效果。然而，使用者運用空間填充曲線生成藝術風格是很困難的。本論文提出使用空間填充曲線自動生成圖像藝術風格的方法。本論文的方法根據圖像特徵，像是灰階值(Grey-level)和梯度(Gradient)等等，控制空間填充曲線在圖像中不同區域的分割。我們使用三種空間填充曲線，即高斯帕曲線(Gosper curve)，皮亞諾曲線(Peano curve)，高斯帕二次曲線 quadratic Gosper curve)生成圖像藝術風格。對輸入圖像直接引入一個疊代技巧，產生以線條為基礎的藝術品。所提方法能保持色調，並取得理想的效果。

An Artistic Rendition of Images Based on Space-filling Curves

Student: Ya-Ying Liao Advisor: Prof. Zen-Chung Shih

Prof. Chin-Chen Chang

Institute of Multimedia Engineering

National Chiao Tung University

ABSTRACT

Space-filling curves have been used for artistic presentation for a long time. However, it is difficult for users to use space-filling curves to generate an artistic representation of an image. In this thesis, we present an approach using space-filling curves to produce artistic styles of images automatically. The proposed approach controls subdivision of a space-filling curve according to image features, such as gray-levels and gradients. We use three kinds of space-filling curves, Gosper curve, Peano curve, and quadratic Gosper curve, to generate artistic styles of images. We introduce an iterative technique for input images to produce line-based drawings directly. The proposed approach can preserve tone and obtain desired results.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisors, Prof. Zen Chung Shih and Prof. Chin-Chen Chang for their guidance and patience. Without their encouragement, I would not complete this thesis. Thanks also to all the members in Computer Graphics and Virtual Reality Laboratory for their reinforcements and suggestions. I deeply appreciate my friend Yu-Chen Wu who discussed with me when I was confused. I am also grateful to Te-Mei Wang who gave me many suggestions on my thesis. Thanks to my teammates in the school table tennis team for their company when I was in pressure. Finally, I want to dedicate the achievement of this work to my family.

Contents

List of Tables

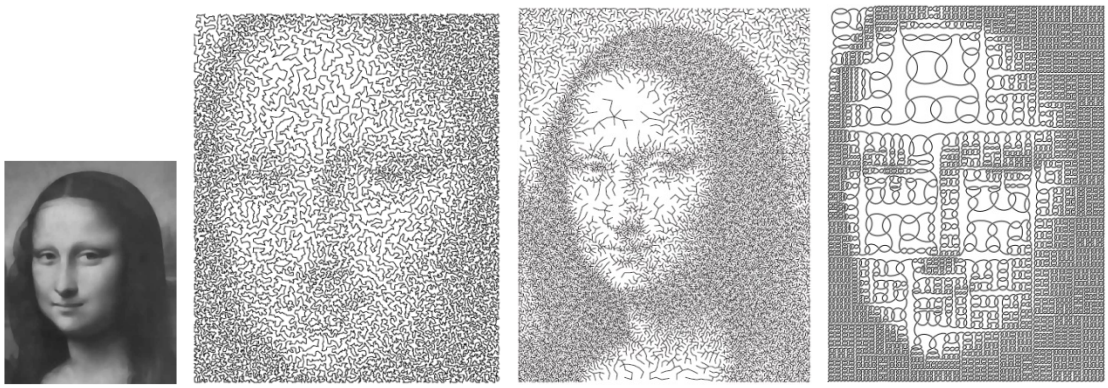
List of Figures

找不到圖表目錄。

Chapter 1

Introduction

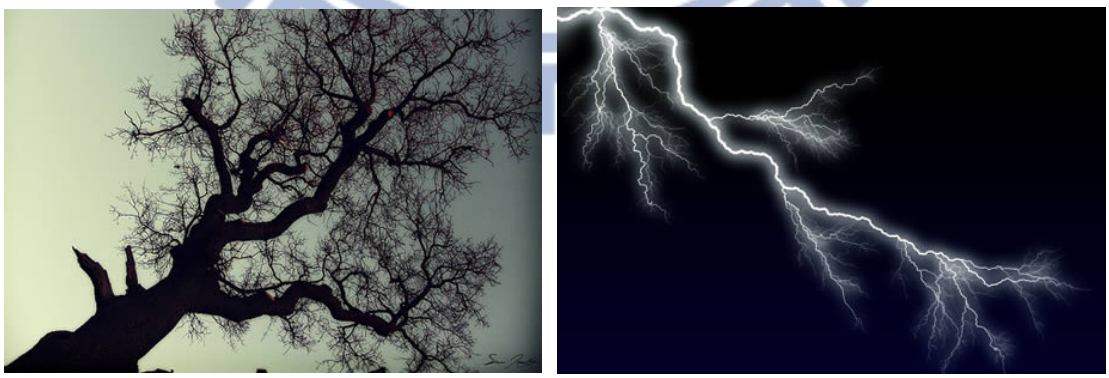
Non-photorealistic rendering (NPR), unlike the traditional rendering focusing on photorealism, is inspired by artistic styles such as painting, drawing, technical illustration, and animated cartoons. One of the most popular methods is the line-based rendering, which generates artistic lines to produce the final painting. There are many kinds of line-based rendering. Bosch and Herman [3] used the traveling salesman problem (TSP) to yield continuous line drawings of target pictures, as shown in Figure 1.1(b). Inoue and Urahama [11] generated a stippling from an image first and then connected all the dots in the stippling with a minimum spanning tree to generate the halftoning image, as shown in Figure 1.1(c). Ahmed [1] first presented an approach to divide a grayscale image into rectangles with equal amount of ink. Then the resulting structure is used to generate novel line-based halftoning techniques, as shown in Figure 1.1(d). For another kind of curves—fractal curves [24], they can improve the traditional NPR method to a new kind of line-based artistic drawing.



(a) (b) (c) (d)

Figure 1.1: Examples of line-based halftoning artworks. (a) Input image, (b) Bosch and Herman [3], (c) Inoue and Urahama [11], and (d) Ahmed [1].

A fractal is a never-ending pattern that repeats itself at different scales. They are generated by repeating a process in a feedback loop. In nature, fractal is everywhere. Examples include trees, lightening, coastline, snowflake, etc., as shown in Figure 1.2. All of these organic structures demonstrate a need to cover as much space as possible. In math, such as Canor sets, Koch curves, Sierpinski triangles, and space-filling curves are also fractals. It can be generated by a computer calculating a simple equation. In this thesis, we mainly focus on space-filling curves.



(a) (b)

Figure 1.2: Fractals in nature. (a) Fractal tree (by Samuel Judge). (b) Lightening fractal (by howstuffworks). [25]

Space-filling curves have attracted both artists and mathematicians. A space-filling curve is defined by passing through every point of a given region. It has been applied in other image processing tasks, such as halftoning [5, 22] and texture analysis [12, 20]. Due to the adjacency property of the space-filling curve, Zang et al. [27] employed it for structure-aware image smoothing. Artists such as Happersett [8] have used space-filling curves and there are many related books and web sites [14, 23, 24], as shown in Figure 1.3. Wyvill [26] developed a method for painting space-filling curves. They called the system, FlowPaint. Users can enjoy the painting on the system. However, this designed system focuses on user interaction. Artists require a certain ability to draw.

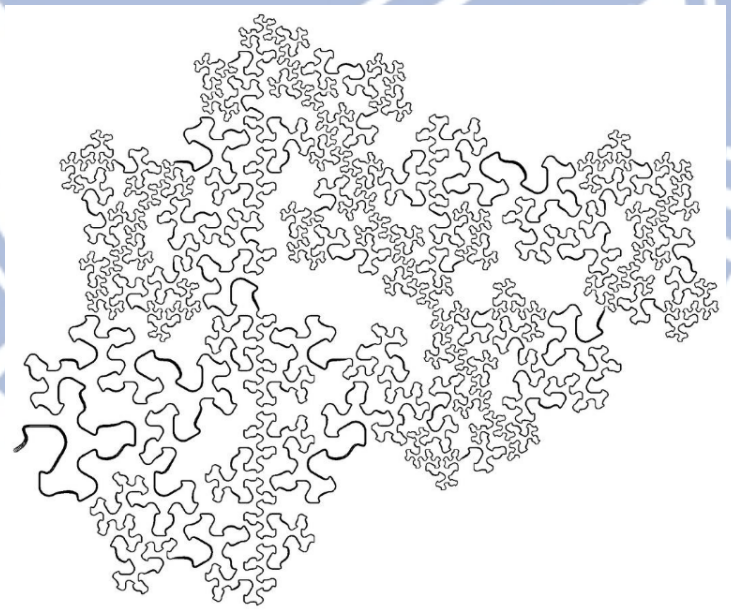


Figure 1.3: Fractal curve [23]

In this thesis, we propose a novel approach to generate artistic styles of images based on space-filling curves automatically. Three kinds of space-filling curves,

Gosper curve, Peano curve, and quadratic Gosper curve, are used in our method. For each curve, we first compute a gray image from an input color image. For each part of the image, the deeper the color, the higher the breakdown level. After using the bilateral texture filter for image smoothing, we use edge detection to detect edges. The subdivision levels of edges of the image are also high. We combine local average gray value and edge density to find split thresholds. We also apply the recursive subdivision method [26] to create a space-filling curve art.

The contributions of this thesis are summarized as follows:

- We propose an automatic method that subdivides the space-filling curve non-homogeneously.
- We combine two image features, namely, gray-levels and gradients, to decide split thresholds.
- We design two new numbering rules for Peano curves and quadratic Gosper curves.

The rest of this paper is organized as follows. Chapter 2 briefly reviews related works. In Chapter 3, we introduce the proposed approach and describe the detail of each step. Chapter 4 shows the results. Conclusions and future works are given in Chapter 5.

Chapter 2

Related Works

2.1 Halftoning

Halftoning is a method for rendering a continuous tone image in a monochrome device. The objective of halftoning is to make the rendered image look like the original one. Ahmed [1] observed that the general paradigm to all stylized halftoning approaches is “a close-up look at the rendered image reveals unexpected details”, where expected details are screen pixels and meaningless printer dots, and unexpected details include letters [15], detailed shapes [16], Truchet tiles [4], solid circles [21], and so on. In this work, unexpected details should be parts of space-filling curves, as shown in Figure 2.1. We set up different depth levels at different parts of an image, which leads to a continuous tone.

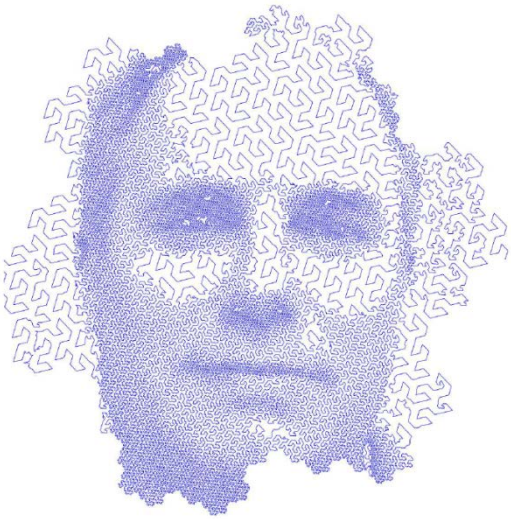


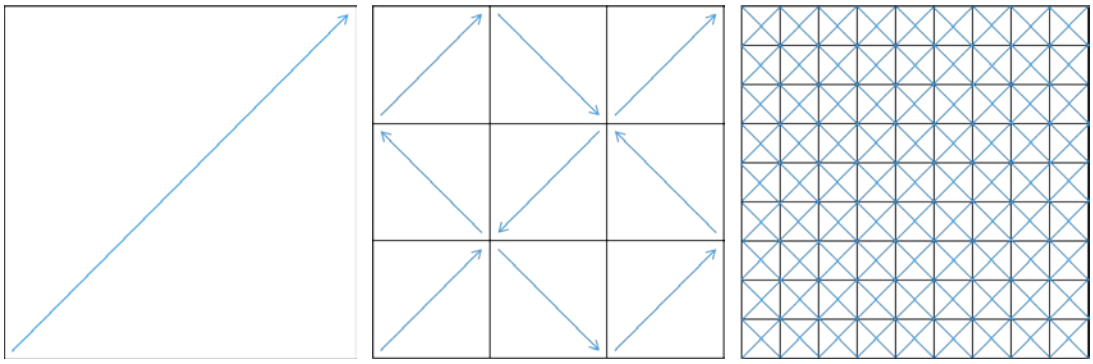
Figure 2.1: Space-filling curve halftoning [26].

2.2 Space-filling Curves

In mathematical analysis, the range of a space-filling curve contains the whole two-dimensional unit square. In 1890, Peano was the first to find a space-filling curve called the Peano curve [19]. In 1891, Hilbert found another space-filling curve called Hilbert curve [10]. Many other space-filling curves have been found since 1900.

Wyvill [26] showed that there is a cellular structure to the curve. All cells have input and output points. A cell is replaced by subdivided cells at the next level of recursion. The input and output points of the cell are identical to those in the next level. When connecting these points, it leaves a trail to form the space-filling curve automatically.

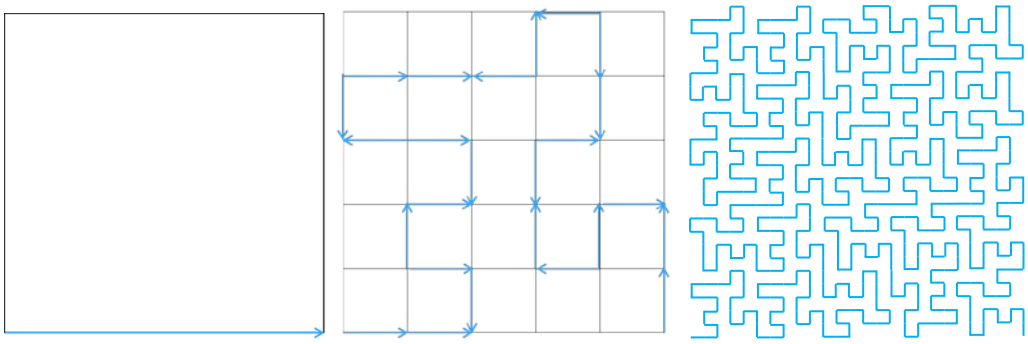
There is a square-based cell for constructing the Peano curve [19]. It starts from the diagonal line of a square-based cell, as shown in Figure 2.2(a). When the unit square is divided into 9 small squares, shrink the previous curve to one third of its original size. Simultaneously, decrease the square size by the factor of three. Nine squares with the direction of each line indicated by arrows are shown in Figure 2.2(b). When all squares are divided into the next level, eighteen squares and the curve are shown in Figure 2.2(c).



(a) (b) (c)

Figure 2.2: The first three stages in the generation of Peano curve. (a) Level-0 with the start point is in the bottom-left corner and the end point is in the top-right corner, (b) Level 1 in the direction indicated by arrows, (c) Level 2.

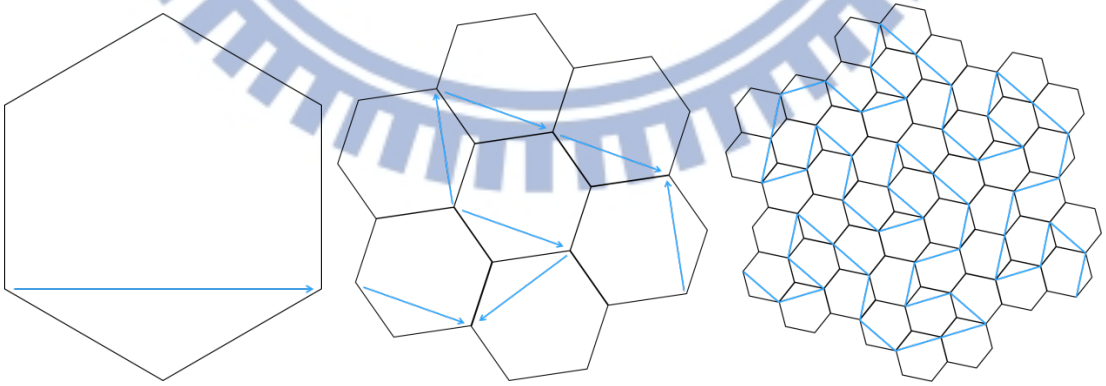
The quadratic Gosper curve [8] is similar to the Peano curve. It also has a square-based cell for constructing the quadratic Gosper curve. At level 0, a line connecting two vertices of a square where the blue arrow indicates the direction, as shown in Figure 2.3(a). When dividing to level 1, the square is divided into 25 small squares. And the line is replaced by 25 scaled down versions of itself and connect points on the subdivided squares, as shown in Figure 2.3(b). The level 2 construction of the curve is shown in Figure 2.3(c).



(a) (b) (c)

Figure 2.3: The first three stages in the generation of quadratic Gosper curve. (a) Level 0 with the start point in the bottom-left corner and the end point in the bottom-right corner, (b) Level 1 in the direction indicated by arrows, (c) Level 2.

Unlike these two curves, the Gosper curve [26] has a hexagon-based cell, as shown in Figure 2.4. At level 0, a blue line connects two vertices of the hex skipping a single vertex. To progress from level-0 to level-1, the line is replaced by 7 scaled down versions of itself and connect points on the subdivided hexagons. The cellular structure is used in our algorithm.



(a) (b) (c)

Figure 2.4: The construction of Gosper curve. (a) Level 0, (b) Level 1, and (c) Level 2.

Chapter 3

The Algorithm

3.1 System Overview

We develop an automatic algorithm for generating artistic rendering of images in a non-homogeneous manner. Space-filling curve is the basic element to construct the artwork. The concept of non-homogeneous manner is doing recursion locally. Every part of a contour should form parts of a space-filling curve.

The system overview is shown in Figure 3.1. First, we compute a gray-level image from an input color image and perform histogram equalization to normalize the image, as shown in Figures 3.1(b) and 3.1(c). To prevent from messy textures and find image structures, we use the bilateral texture filter [13] to smooth the input image, as shown in Figure 3.1(d). Then, we use Canny edge detection [6] to detect edge pixels, as shown in Figure 3.1(e). According to our observations, for Gosper curve and Peano curve, maximum depth 6 is dense enough to express the halftone. For quadratic Gosper curve, maximum depth 4 is enough. For each curve, if the maximum depth is n, we need to decide n-1 split thresholds. According to local intensity level and the number of edge pixels, n-1 thresholds are set. By dividing level 1 to level n recursively, we get various cells from level 1 to level n, as shown in Figure 3.1(f). With the cell numbering rule for Gosper curve introduced by Wyvill [26], we number cells in Peano curve and quadratic Gosper curve in a similar manner. Then, we use the method [26] to connect all input and output vertices of cells. Finally, we obtain the line-based

drawing result by combining all steps above, as shown in Figure 3.1(g).

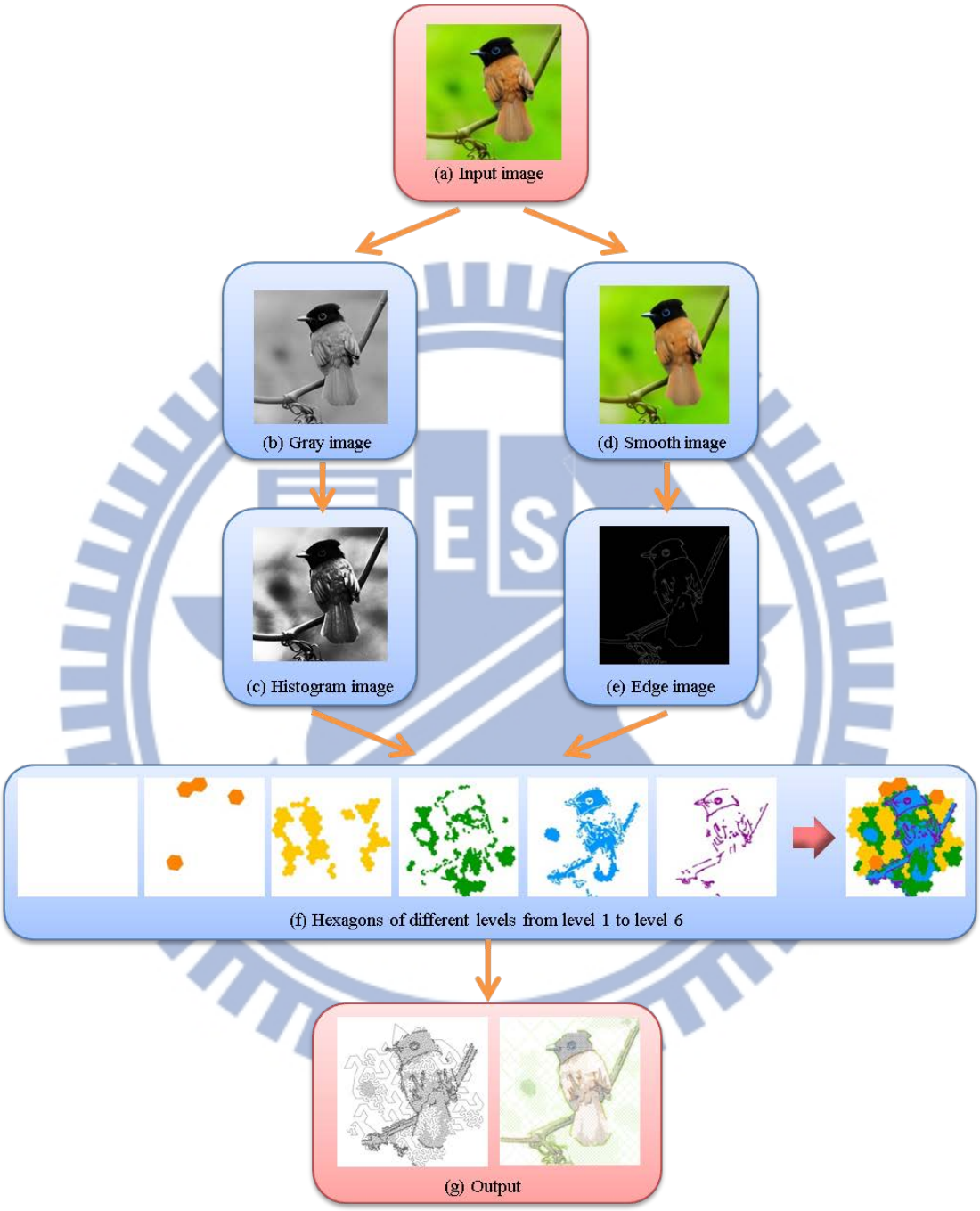


Figure 3.1: System overview (Gosper curve)

3.2 Gray-Level Features

Gray-level is one of the most important primitive features for image understanding. It affects recursive levels a lot. Low gray values usually correspond to deep depth level for better effect. In a color image, each pixel has RGB values. In order to compute gray levels, we first use equation (1) [18] to transfer an input color image to a grayscale image.

$$\text{Gray} = 0.299 * \text{Red} + 0.587 * \text{Green} + 0.114 * \text{Blue} \quad (1)$$

Histogram is a graphical representation of the intensity distribution of an image. Histogram equalization is a technique for adjusting image intensities to enhance contrast, and make the histogram of the image more even. If the original image has uneven distribution of color, the local depth level of the curve is unclear, which makes the result blurring. To enhance image quality and for further handling on images, histogram equalization is required.

Consider a discrete grayscale image $\{x\}$ and let n_i be the number of pixels of gray level i. The probability of an pixel belongs to level i in the image is

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \ 0 \leq i < L \quad (2)$$

where L is the number of possible intensity values, often 256. n is the total number of pixels in the image. $p_x(i)$ is the image's histogram for pixel value i, normalized to $[0..1]$. The *cumulative distribution function* (CDF) corresponding to p_x is

$$CDF_x(i) = \sum_{j=0}^{i} p_x(j) \qquad (3)$$

We would like to create a transformation of the form $y = T(x)$ to produce a new image $\{y\}$. The histogram of image $\{y\}$ is more flat.

$$y = T(x) = CDF_x(i) * (L-1),\ 0 \le i < L \qquad (4)$$

For example, an input color image, Figure 3.2(a), is transformed to a grayscale image, Figure 3.2(b). After doing histogram equalization, the result is shown in Figure 3.2(c).

(a) (b) (c)

Figure 3.2: Image processing. (a) Input color image, (b) Gray scaling, (c) Histogram equalization.

3.3 Edge Features

Edges are defined in an image as the points with sharp changes of color or intensity, while suggesting contours of significant subregions. They are thus another important primitive feature for image understanding. Cells close to edges have deeper depth level than those far from edges. Edge detection is the most common approach for detecting meaningful discontinuities in gray level. Some popular edge detection techniques involve masking based operators like Roberts, Sobel, Prewitt, etc. Roberts edge detection operator uses local differentiation to detect an edge. It is sensitive to noise. Sobel edge detection operator uses gradients of neighbor pixels to compute one pixel's gradient. But it does not separate the topic of the image and background strictly because its processing does not base on gray level. Also, Laplacian of Gaussian (LoG) is a useful method for edge detection which works on the second derivative of the image. But it is hard to adjust parameters to adapt every image. Canny edge detection algorithm [6] is a classical and robust method for edge detection in grayscale images. It works on three main criteria: low error rate, accurate localization of edge points and one response to a single edge. In this thesis, we select the Canny edge detector [6] to find image edges.

Most of the surface of an object is not smooth and contains noises. If we apply the Canny edge detector [6] directly on input images, the detector may confuse textures with edges. Here, we just need to extract the main image structure. Therefore, we need to use edge preserving filter to remove image details without blurring edges before doing edge detection. Lin's bilateral texture filter [13] is a good choice because it can remove image noises and preserve image edges simultaneously. It improves bilateral texture filtering algorithm [7] by adjusting patch sizes. As long as the input

image is smoothed by the filter, the edge detection result will be clearer. Figure 3.3 shows the comparison of results between filtering and without filtering.

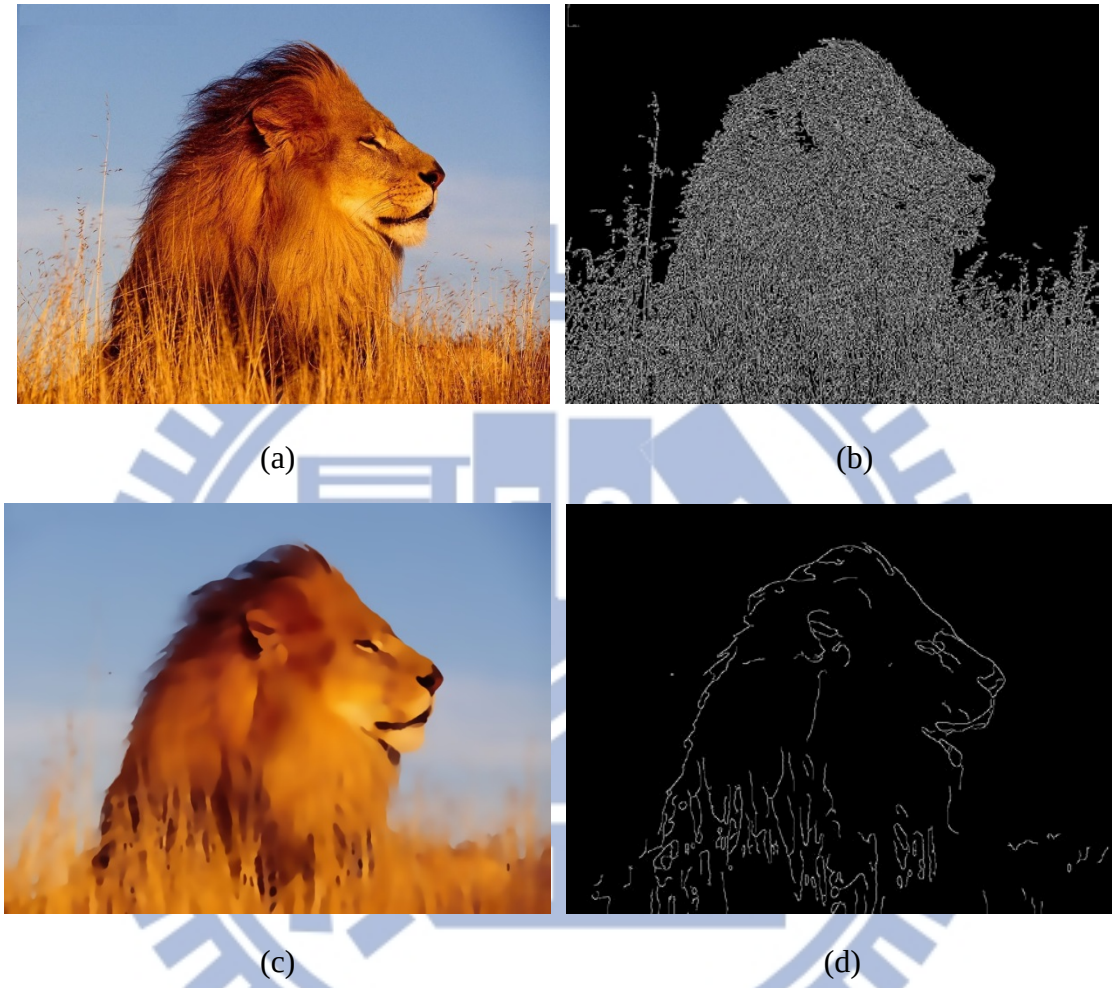


Figure 3.3: Bilateral texture filtering. (a) Input image, (b) Result of (a) by Canny edge detection, (c) Result of (a) by bilateral texture filter, (d) Result of (c) by Canny edge detection.

3.4 Thresholding

After getting two important features, we need to set thresholds for subdivision. For each curve, if the maximum depth is n, we need to decide n-1 split thresholds. For example, the maximum depth of Gosper curve is 6. There are 5 thresholds based on gray-level and gradient features.

In terms of gray values, we use the improved version of Otsu's method [17]. Otsu's method is used to automatically perform clustering-based image thresholding, or the reduction of a gray level image to a binary image. It calculates the optimum threshold separating the image into two classes so that their intra-class variance is minimal, and their inter-class variance $\sigma_b^2(t)$ is maximal, as shown in equation (5).

$$\sigma_b^2(t) = P_1(t)P_2(t)[\mu_1(t) - \mu_2(t)]^2 \qquad (5)$$

where $P_1(t)$ is the probability of gray levels smaller than t, and $P_2(t)$ is the probability of gray levels larger than t, that is $1 - P_1(t)$. $\mu_1(t)$ is the mean of gray levels smaller than t, and $\mu_2(t)$ is the mean of gray levels larger than t. We need to find a t from 1 to 254 such that equation (5) is maximized.

Instead of finding a threshold in Otsu's method, we iterate this method 3 times to find 3 thresholds. After we first calculate a threshold like the original Otsu's method, the image is clustered into 2 classes. Then we use Otsu's method for these 2 classes, respectively. And then plus 0 and 255 for thresholding. For each curve, we put these 5 thresholds in descending order. Start from level-1, each threshold is set to 255.

Edge detection enables us to understand edge pixels distribution. At each level, we compute the average number of edge pixels inside all cells. This value is treated as the edge threshold between the current level and the next level.

3.5 Recursive Division

We show how to divide cells in non-homogenous manner under the two features described above. Each cell is looked as a patch and we compute data inside each patch and decide whether to divide it or not. For Gosper and Peano curves, their cell structures are hexagons. For quadratic Gosper curve, its cell structure is a square. First, put a cell structure on the center of an image, it is known as level 0 patch. The size of the level 0 patch is maximized but smaller than the image, covering the whole scene as much as possible. Starting from level 1, we compute an intensity level and the number of edge pixels in each patch. The intensity level $I(P)$ in patch P is computed by equation (6)

$$I(P) = \sum_{x \in P} \frac{I(x)}{n(P)} \qquad (6)$$

where $I(x)$ is the intensity level of pixel x, $n(P)$ is the number of pixels in patch P. If $I(P)$ is smaller than gray threshold t computed above or the number of edge pixels in patch P is larger than the edge threshold, the corresponding patch is subdivided to the next level, which is known as level 2. For those level 2 patches, we compute an intensity level and the number of edge pixels repeatedly and subdivide patches recursively until the maximum level is reached.

For example, take Figure 3.4(a) as an input image. Its all 5 gray thresholds are 255, 192, 125, 63, and 0 from level 1 to level 5. Its 5 edge thresholds are 1076, 153, 24, 5, and 1 from level 1 to level 5. All level 1 patches are divided down because their $I(P)$ is smaller than 255 or their number of edge pixels is larger than 1076, as shown in Figure 3.4(b). Orange hexagonal patches in Figure 3.4(c) are not divided down because their $I(P)$ is larger than 192 and their number of edge pixels is smaller than 153, which means that colors in these parts are light and there are less edges in these parts. Hexagonal patches from level 3 to level 6 are shown in Figures 3.4(d) – 3.4(g). We can observe that when the level gets deeper, the contour of the bird is getting clear. After getting all separate patches from level 1 to level 6, we overlap them to generate a new structure, as shown in Figure 3.4(h). We use this structure to render the space-filling curve.

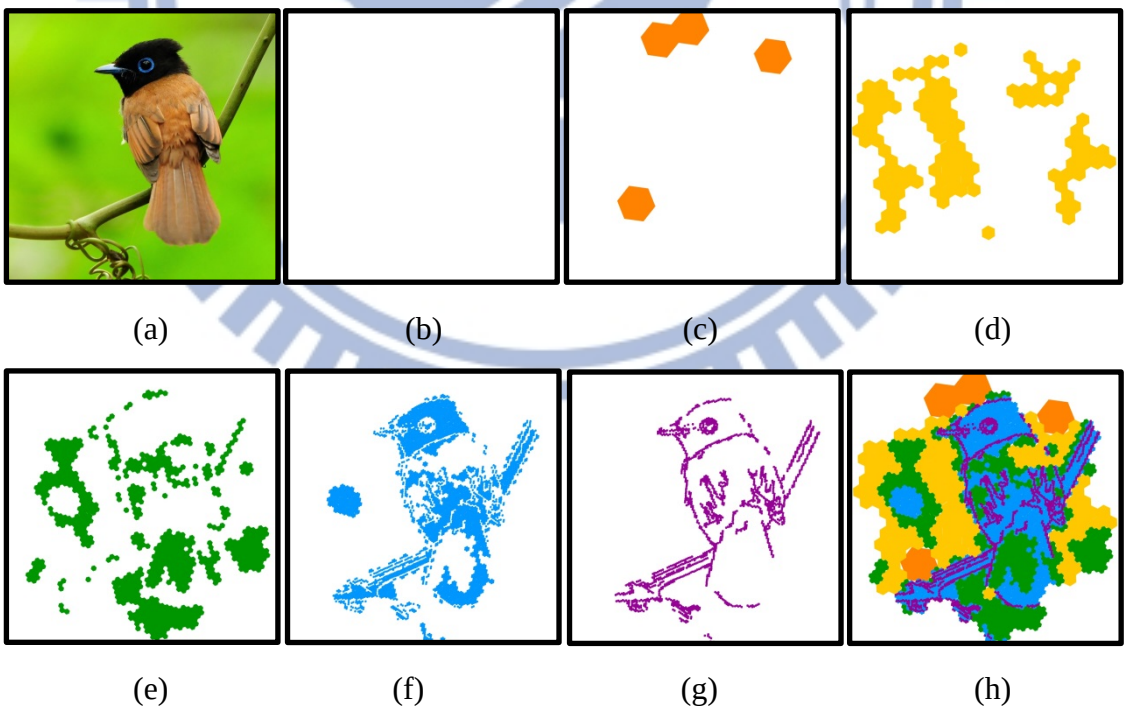


Figure 3.4: Patch division. (a) Input, (b) – (g) Division patches from level 1 to level 6. Notice that there are no level 1 patches in (b) because all level 1 patches are divided down. (h) Overlapping from (b) to (g), we later use this structure for curve rendering.

3.6 Rendering Space-filling curves

We mainly use the structure like the image shown in Figure 3.4(h) to render space-filling curves. Every patch in each level has input and output vertices. By connecting all input and output vertices in each patch, a space-filling curve pattern is shown. For Gosper curve, Wyvill [26] introduced a method for finding input and output vertices in each patch. For Gosper curve, when a patch is divided, we use Wyvill's method [26] to number patches, save types, and find input and output vertices of each patch. For Peano curve and quadratic Gosper curve, we introduce numbering rules for these curves in a similar manner.

For Peano curve, level 0 patch is constructed so that vertex 0 is the lower-left corner vertex, and other 3 vertices are numbered in counterclockwise order. When level 0 patch is divided to level 1, the patch in the center is always 0, and the patch that shared vertex 0 of the parent patch is 1. Then the other 7 patches are numbered in counterclockwise order, as shown in Figure 3.5.

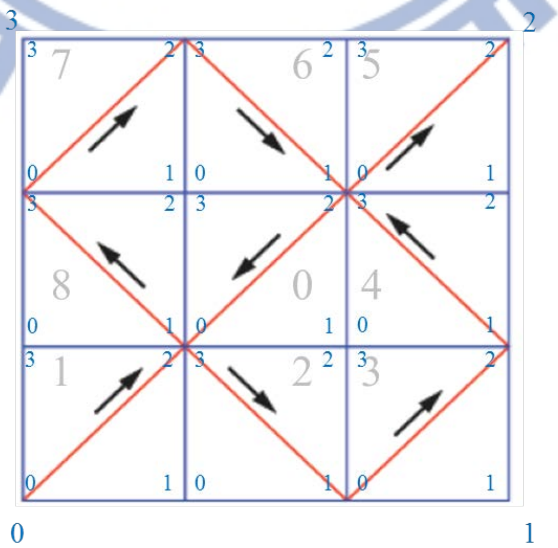


Figure 3.5: Peano curve numbering.

For quadratic Gosper curve, level 0 patch is constructed so that vertex 0 is the lower-left corner vertex, too. The other 3 vertices are numbered in counterclockwise order. When level 0 patch is divided to level 1, the patch in the center is always 0, and the patch which has vertex in common with vertex 0 of the parent patch is 1. Then the other 7 patches are numbered in counterclockwise spiral order, as shown in Figure 3.6.

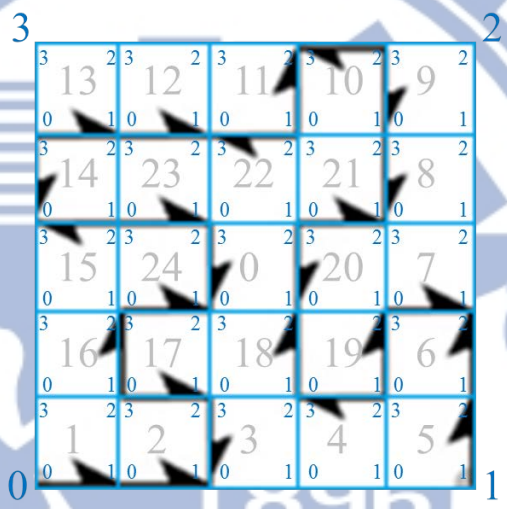


Figure 3.6: Quadratic Gosper curve numbering.

According to these numbering rules, Wyvill's method [26] can be effectively utilized to find out input and output vertices. There are three main steps to draw the curve and three tables are used. The first table is t-table. It stores the type of the curve. Peano curve does not need to use the t-table because it does not have types. For quadratic Gosper curve, if the parent type is forward, types of child patches are labeled as 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 1 1 taken from t-table. Else if the parent type is backward, types of child patches are labeled as 0 0 1 1 0 0 1 0 1 1 0 1 0 1 1 0 1 0 0 1 1 0 0 1 1. When the level is broken down and child patches become

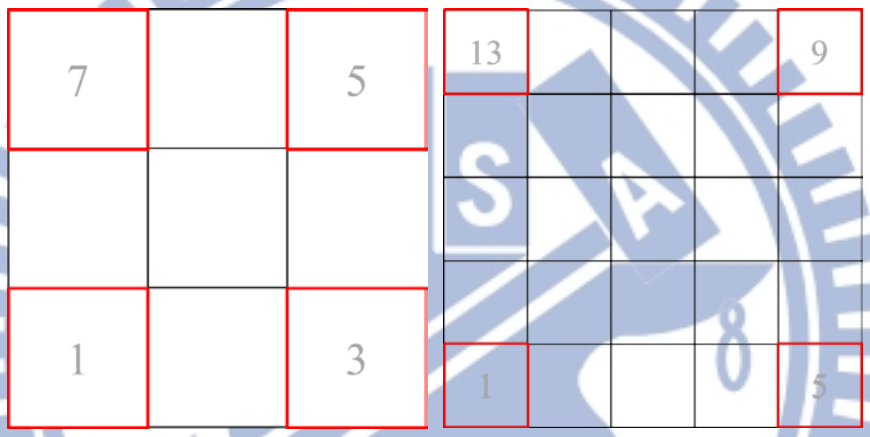
parent patches, their types are taken from the t-table to be parent types. The second table is called s-table, it stores the square patch order along the path of curves. It is calculated by parent type, input vertex, and output vertex. Wyvill [26] proposed an algorithm to order patches for Gosper curve. For Peano curve, we use algorithm 1 to order patches in the s-table.

Algorithm 1: Finding 9 child patches along the Peano curve

Input: Invtx
Output: $s[0], s[1], s[2], \ldots, s[8]$

```
1.   s[0] ← (Invtx * 2) + 1
2.   if s[0] == 1 then
3.     for (i = 1; i < 4; i++) do
4.       s[i] ← (s[0] + 8 − i ) MOD 9
5.     end for
6.     s[4] ← 0
7.     for (i = 5; i < 9; i++) do
8.       s[i] ← (s[0] + i − 4 ) MOD 9
9.   end for
10.  else if s[0] == 3 then
11.    for (i = 1; i < 3; i++) do
12.      s[i] ← (s[0] + 9 − i ) MOD 9
13.    end for
14.    s[3] ← 8
15.    s[4] ← 0
16.    for (i = 5; i < 9; i++) do
17.      s[i] ← (s[0] + i − 4 ) MOD 9
18.    end for
19.  else if s[0] == 5 then
20.    for (i = 1; i < 4; i++) do
21.      s[i] ← (s[0] − i ) MOD 9
22.    end for
23.    s[4] ← 0
24.    for (i = 5; i < 8; i++) do
25.      s[i] ← (s[0] + i − 4 ) MOD 9
26.    end for
27.    s[8] ← 1
28.  else if s[0] == 7 then
29.    for (i = 1; i < 4; i++) do
30.      s[i] ← (s[0] − i ) MOD 9
31.    end for
32.    s[4] ← 0
33.    s[5] ← (s[0] + i − 4 ) MOD 9
34.    for (i = 6; i < 9; i++) do
35.      s[i] ← ((s[0] + i − 4 ) MOD 9) + 1
36.    end for
```

The first square patch s[0] is calculated by input vertex number $Invtx$. There are only four possible values 1, 3, 5, 7 for s[0], which are four red squares in four corners, as shown in Figure 3.7(a). Next 8 squares s[1] to s[8] are calculated by s[0], like step 3 to step 9, step 11 to step 18, step 20 to step 27, and step 29 to step 35. There are also four possible values 1, 5, 9, 13 for quadratic Gosper curve's first entry square, as shown in Figure 3.7(b). By input vertex number and parent type, s-table for quadratic Gosper curve is calculated in a similar manner.



(a) (b)

Figure 3.7: Four possible first entry squares with gray patch number inside them. (a) Peano curve. (b) Quadratic Gosper curve.

Having found ordered numbers of child patches, the next step is to find input and output vertices of child patches. For Peano curve, the output vertex is found by equation (7).

$$Outvtx = (Invtx + 2)\ MOD\ 4 \qquad (7)$$

The output vertex is always the diagonal vertex of the input vertex. For quadratic Gosper curve, output vertex is found by equation (8).

$$Outvtx = \begin{cases} (Invtx + 1)\ MOD\ 4,\ if\ type = 0 \\ (Invtx + 3)\ MOD\ 4,\ if\ type = 1 \end{cases} \quad (8)$$

The output vertex will be affected by the patch type. Due to sharing vertices between child patches, the output vertex of the current patch and the input vertex of the next patch are stored in the table. Finally, from the starting point, connecting all input and output vertices of all patches forms a space-filling curve.

Chapter 4

Results

We run our program on an desktop computer with Intel Core i5 system, 3.00GHz CPU and 8GB memory. Input sources are color images searched from Google engine, and outputs are line-based halftoning images. Input images are smoothed to reflect their structures, as shown in Figure 4.1.

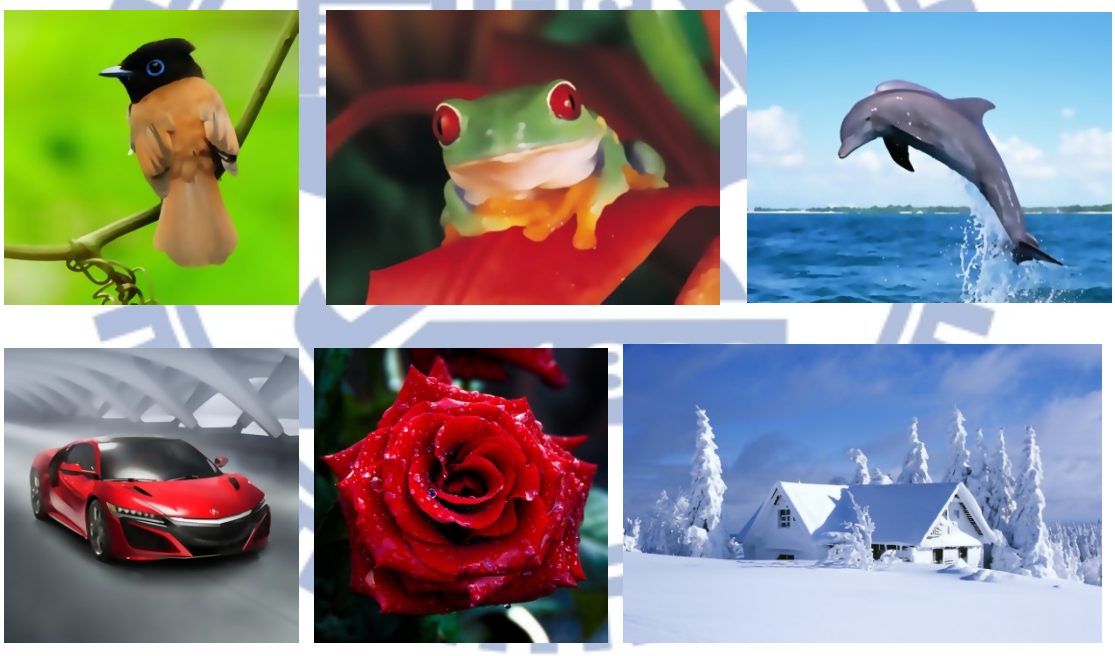


Figure 4.1: Smoothed results of input images.

Figures 4.2 – 4.7 show the results of images using the Gosper curve, the Peano curve, and the quadratic Gosper curve. We first show the histogram equalization result and the edge detection result. These two results are used to find thresholds. The

contour of the main object is extracted first. The three output images are obtained using the three curves.

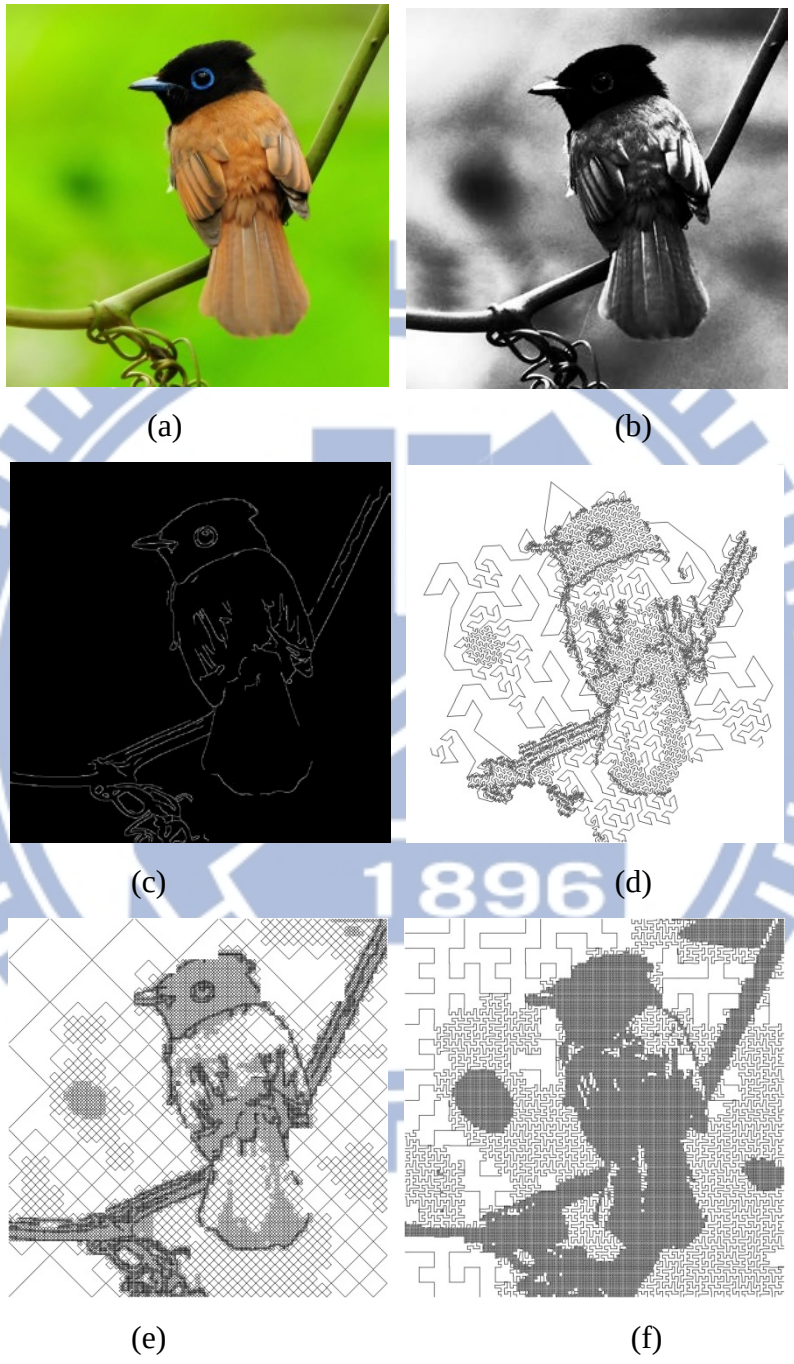


Figure 4.2: Results of *Bird*. (a) Input image, (b) Result of histogram equalization, (c) Result of edge detection, (d) Result using the Gosper curve, (e) Result using the Peano curve, (f) Result using the quadratic Gosper curve.

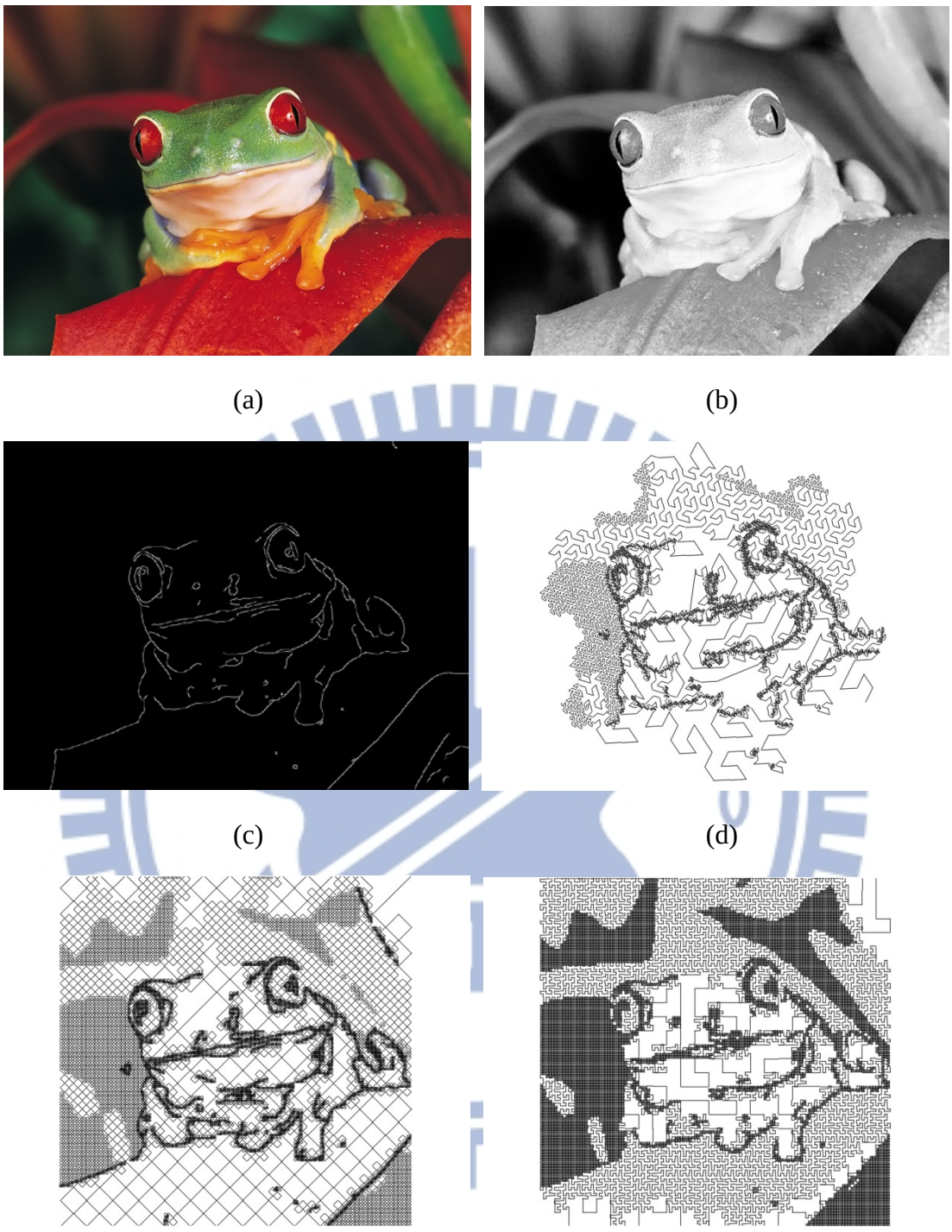


Figure 4.3: Results of *Frog*. (a) Input image, (b) Result of histogram equalization, (c) Result of edge detection, (d) Result using the Gosper curve, (e) Result using the Peano curve, (f) Result using the quadratic Gosper curve.

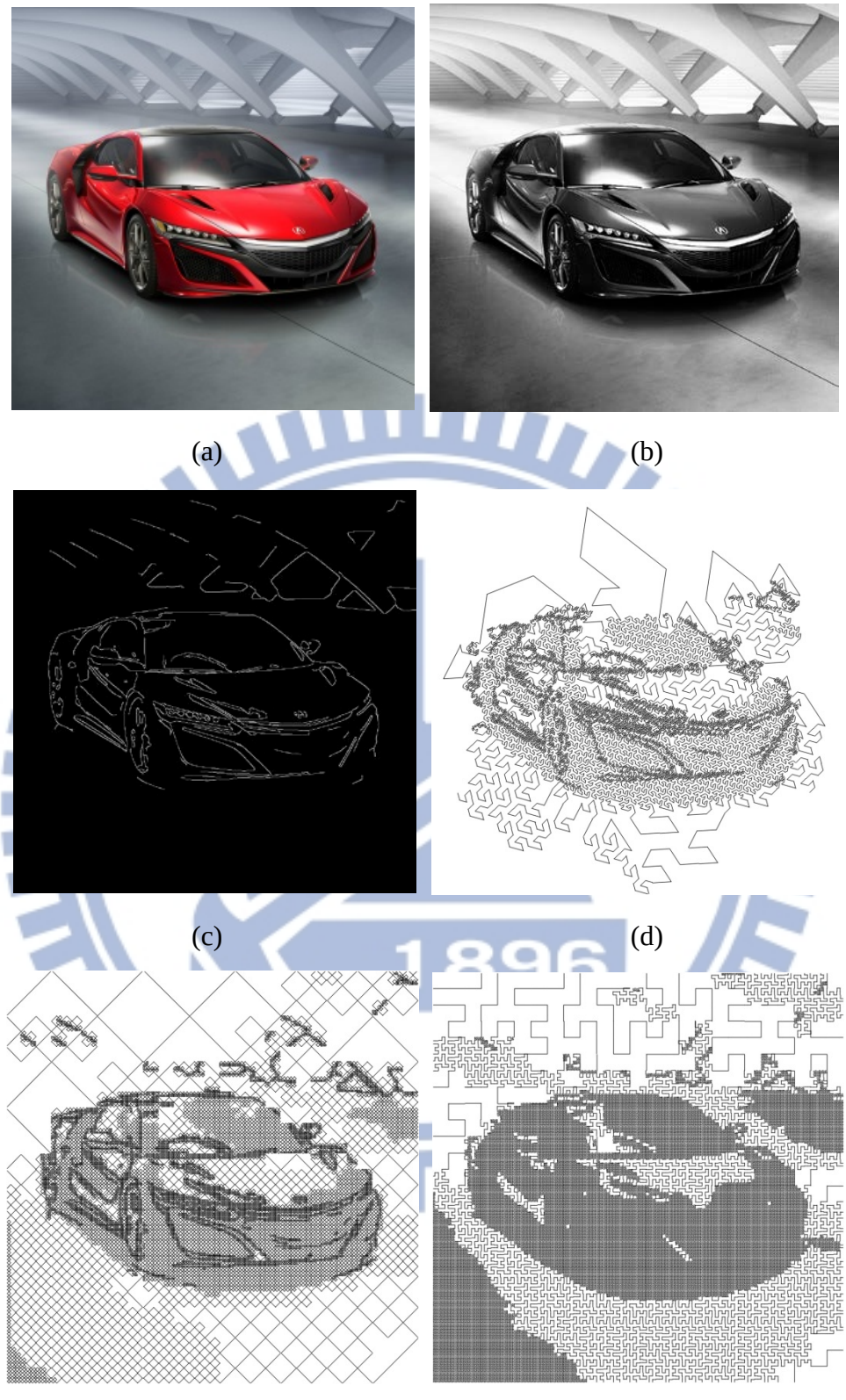


Figure 4.4: Results of *Car*. (a) Input image, (b) Result of histogram equalization, (c) Result of edge detection, (d) Result using the Gosper curve, (e) Result using the Peano curve, (f) Result using the quadratic Gosper curve.

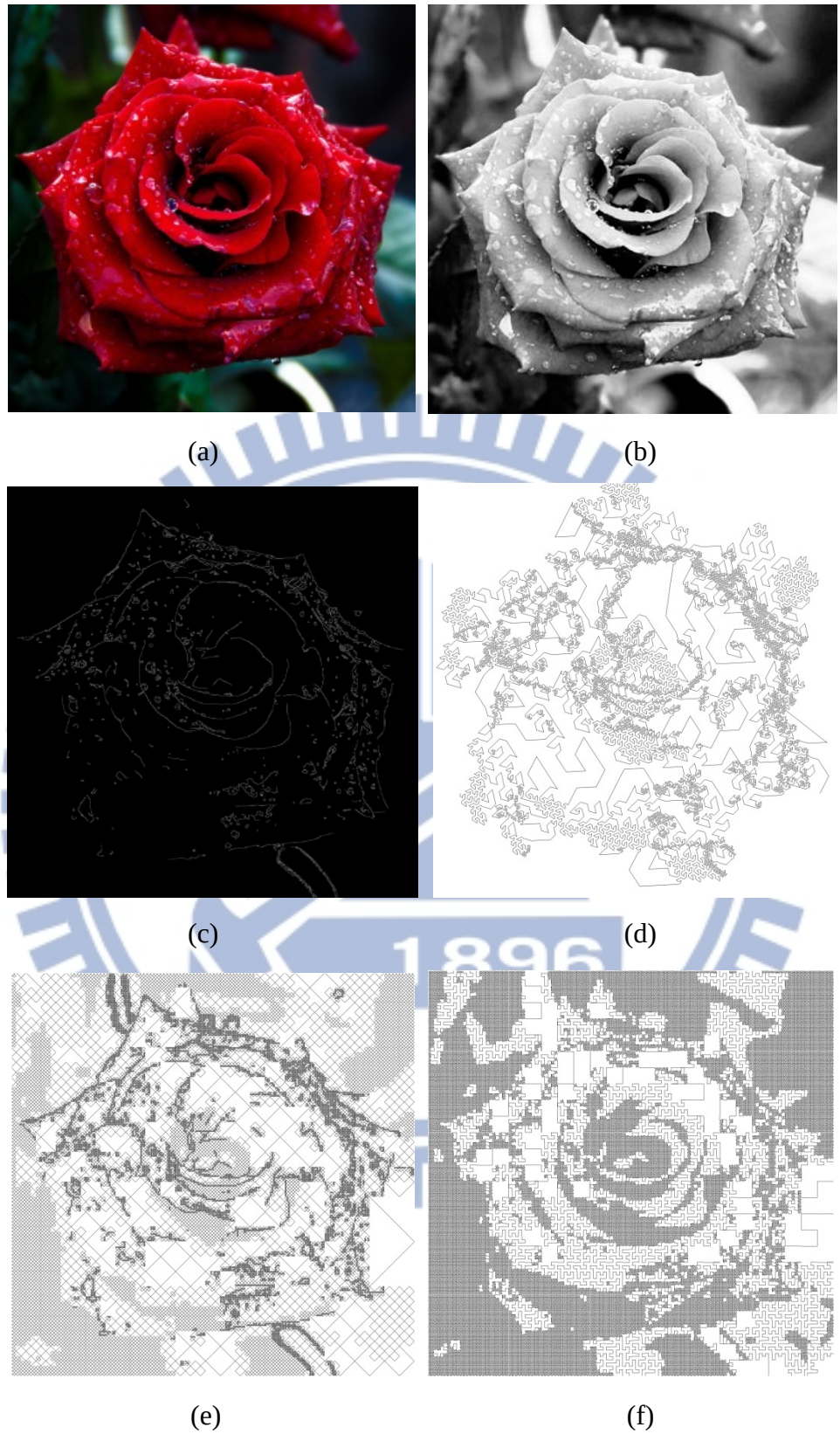


Figure 4.5: Results of *Rose*. (a) Input image, (b) Result of histogram equalization, (c) Result of edge detection, (d) Result using the Gosper curve, (e) Result using the Peano curve, (f) Result using the quadratic Gosper curve.

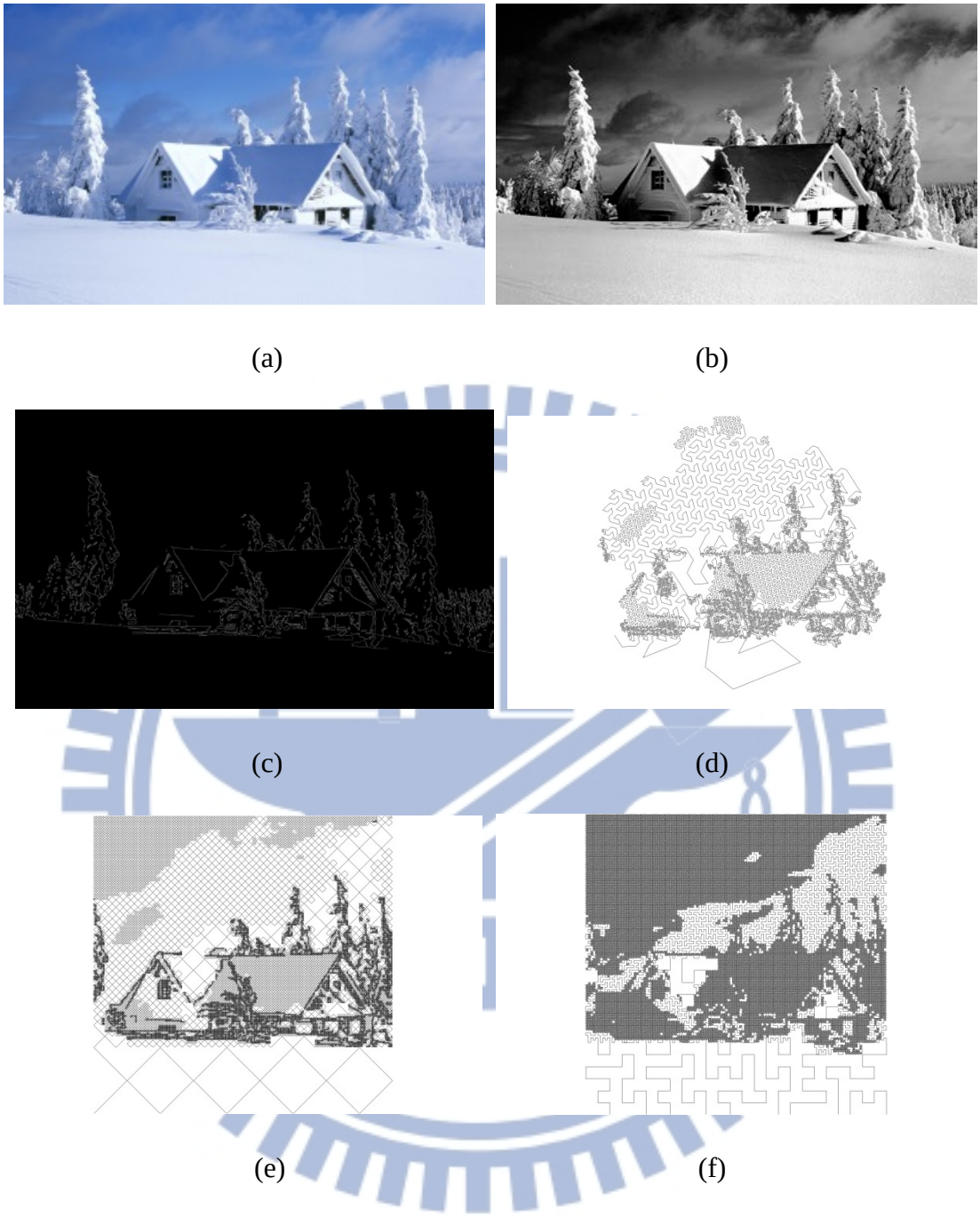


Figure 4.6: Results of *House*. (a) Input image, (b) Result of histogram equalization, (c) Result of edge detection, (d) Result using the Gosper curve, (e) Result using the Peano curve, (f) Result using the quadratic Gosper curve.

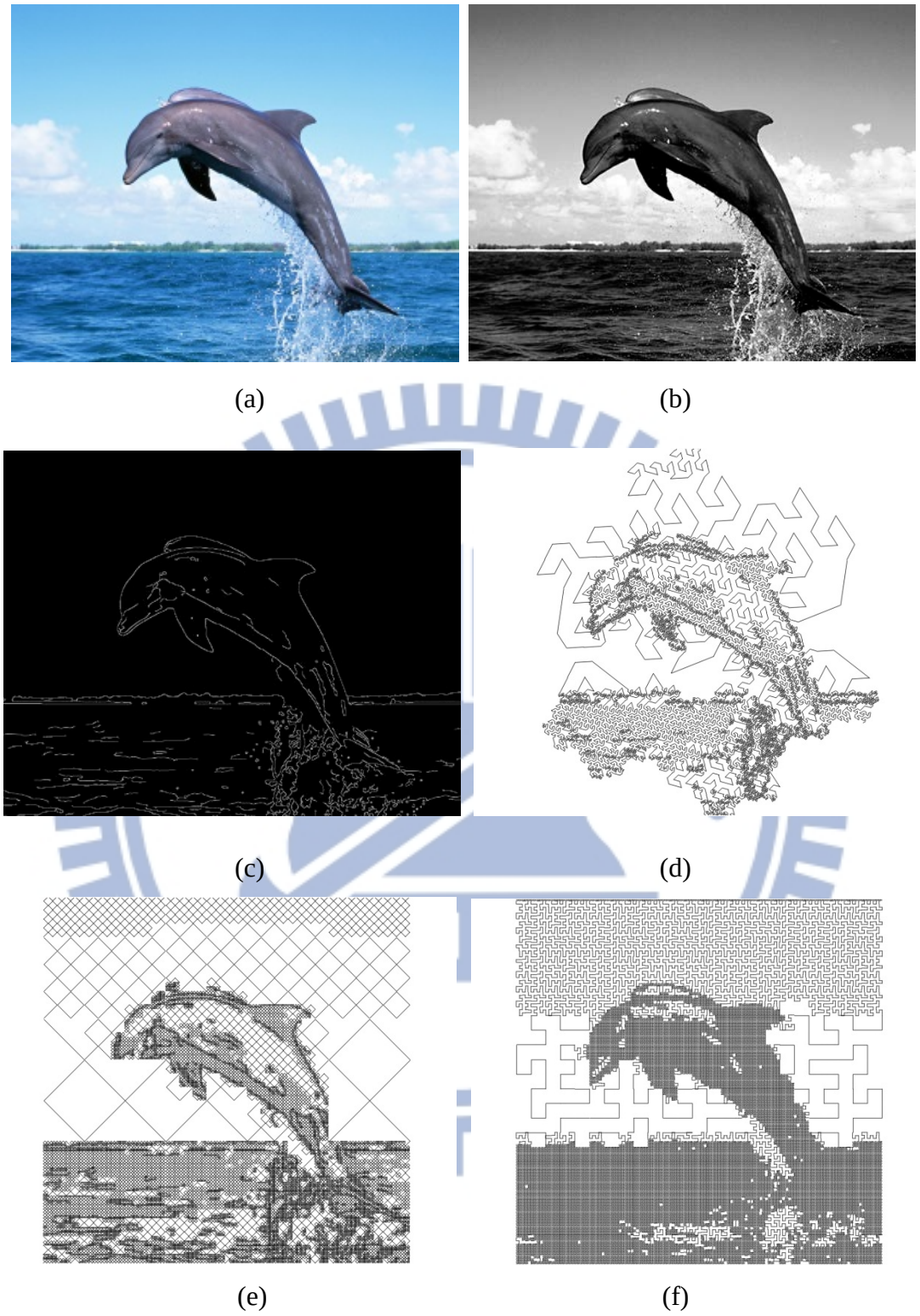


Figure 4.7: Results of *Dolphin*. (a) Input image, (b) Result of histogram equalization, (c) Result of edge detection, (d) Result using the Gosper curve, (e) Result using the Peano curve, (f) Result using the quadratic Gosper curve.

If we apply the original colors of the input image on the halftoning image, another kind of artistic rendition appears, as shown in Figure 4.8.

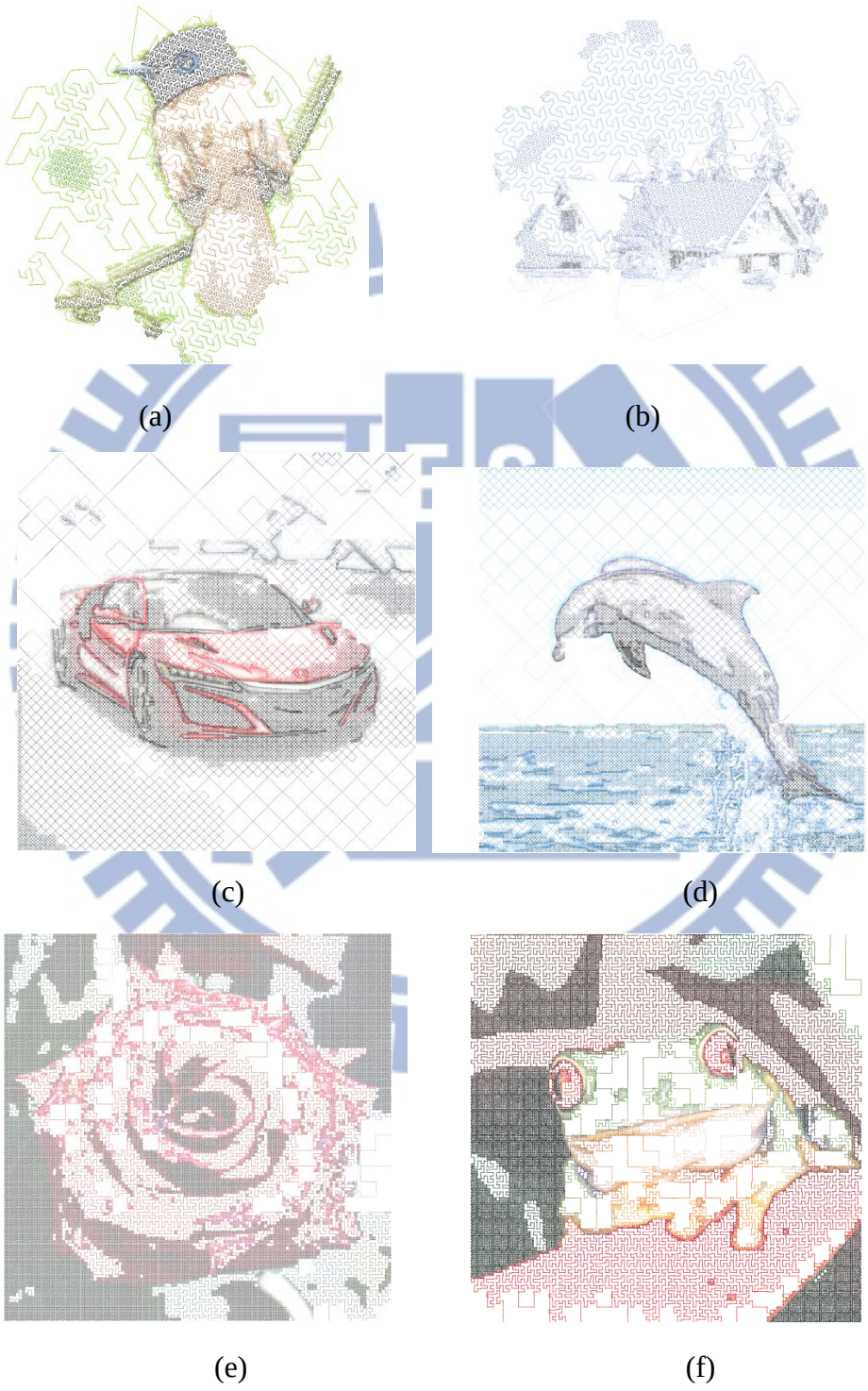


Figure 4.8: Results with colors. (a) and (b) Gosper curve, (c) and (d) Peano curve, (e) and (f) quadratic Gosper curve.

Thresholds for each levels are shown in Table 4.1. There are two kinds of thresholds for each input: gray level and edge pixel. Each kind has 5 thresholds: d_1 to d_5. d_1 means the threshold from level 1 to level 2. d_2 means the threshold from level 2 to level 3, and so on. For Gosper curve and Peano curve, 5 thresholds are used from d_1 to d_5. For quadratic Gosper curve, 3 thresholds are used from d_1 to d_3. No matter what kind of thresholds, the curve gets smaller while the level becomes deeper.

Table 4.1: Thresholds of each input image.

Input	**Thresholds**	d_1	d_2	d_3	d_4	d_5
bird	gray	256	192	125	63	0
	edge	1076	153	24	5	1
car	gray	256	190	127	63	0
	edge	2079	266	46	8	1
frog	gray	256	192	128	65	0
	edge	771	104	15	3	1
dolphin	gray	256	192	128	65	0
	edge	1498	196	33	8	1
house	gray	256	191	127	64	0
	edge	3478	524	84	14	5
flower	gray	256	192	128	65	0
	edge	6390	841	124	21	5

The resolution of input images and execution time of these three curves are shown in Table 4.2. T_g is the execution time of Gosper curve. T_p is the execution time of Peano curve. T_q is the execution time of quadratic Gosper curve. We can observe that the bigger of the input image, the longer of the execution time. T_g is smaller than T_p and T_q because the number of divided patches of Gosper curve is 7, which is smaller than Peano curve and quadratic Gosper curve. Besides, T_p is bigger than T_q because the maximum level of Peano curve is 6, which is bigger than quadratic Gosper curve.

Table 4.2: Execution time of the three curves.

Input	**Resolution**	T_g **(Sec)**	T_p **(Sec)**	T_q **(Sec)**
bird	1024*1024	13	78	17
car	1024*1024	14	96	18
frog	1024*768	9	84	15
dolphin	1280*1024	7	79	15
house	2500*1600	22	95	25
flower	2048*2048	23	99	33

We also compare our method with the method of Bickford et al. [2], as shown in Figure 4.9. They subdivided each section of the fractal using a quadtree, until the desired darkness value was reached. Since we use edge features of the image, the borderline of the person in the image is more obvious than that of the method of Bickford et al., as shown in Figure 4.9(c). Figures 4.9(d) to 4.9(f) show another example. Unlike the unanimous black of the overcoat in Figure 4.9(e), contours of the hat, the glass, and the collar are more clear, as shown in Figure 4.9(f).

(a) (b) (c)

(d) (e) (f)

Figure 4.9: Results and comparisons. (a) and (d) are input images, (b) and (e) are the results of the method of Bickford et al. [2], (c) and (f) are the results of our method.

Chapter 5

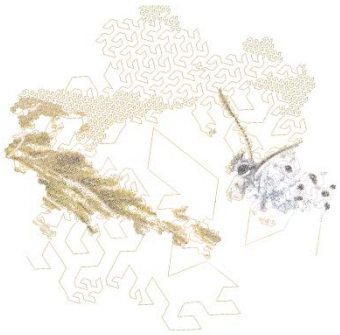
Conclusions and Future Works

In this thesis, we have explored the halftoning art based on space-filling curves, and the methods for achieving desired artistic effects. We use the improved Otsu's method to decide gray level thresholds, and compute average edge pixel to decide edge thresholds. Users do not need to decide thresholds manually. According to image features, cell structures at different levels are integrated to generate a structure image. We create new numbering rules for Peano curve and quadratic Gosper curve rendering. One limitation of our approach is that the main object should be in the center of the image because the level 0 patch is put in the center. Otherwise it would fail to cover the main character, which leads to a unreasonable result.

One of limitations of our method is that level 0 patch should cover all the main object of the image. If the main object is not on the center of the image, parts of it will be cut on the final result. Figure 5.1 shows failed examples. Wings of the butterfly are cut on the final result, as shown in Figure 5.1(b). If the main object is too large and go beyond the scope of the level 0 patch, the final result will not complete, as shown in Figure 5.1(d).

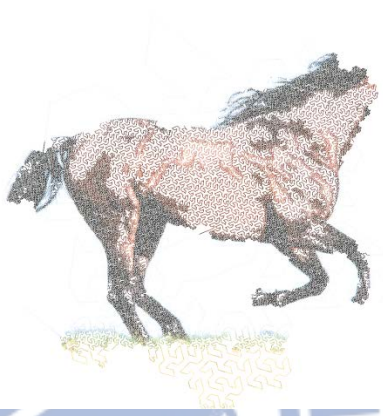
There are some directions for the further research. Except for the three curves mentioned in this thesis, there are many kinds of space-filling curves in the world. The proposed method can be applied to other curves. Besides, the proposed technique can be extended to create animations.

(a)



(b)

(c)



(d)

Figure 5.1: Failed examples. (a) and (c) are input images, (b) is the result of (a), (d) is the result of (c).

Reference

[1] Ahmed, A. G. (2014). Modular line-based halftoning via recursive division. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering*, 41-48. ACM.

[2] Bickford, N. (2013). Hilbert(Hilbert). Retrieved from https://nbickford.wordpress.com/2013/05/19/hilberthilbert/

[3] Bosch, R., & Herman, A. (2004). Continuous line drawings via the traveling salesman problem. *Operations research letters*, *32*(4), 302-303.

[4] Bosch, R. (2011). Opt art: special cases. *Proceedings of Bridges 2011: Mathematics, Music, Art, Architecture, Culture*.

[5] Buchanan, J. W., & Verevka, O. (1995). Edge preservation with space-filling curve half-toning. In *Graphics Interface*. 75-75. CANADIAN INFORMATION PROCESSING SOCIETY.

[6] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6), 679-698.

[7] Cho, H., Lee, H., Kang, H., & Lee, S. (2014). Bilateral texture filtering. *ACM Transactions on Graphics (TOG)*, *33*(4), 128.

[8] Dekking, F. M. (1982). Recurrent sets. *Advances in Mathematics*, *44*(1), 78-104.

[9] Happersett, S. (2015). Artist interview: Irene rousseau. *Journal of Mathematics and the Arts*, *9*(1-2), 37-43.

[10] Hilbert, D. (1891). Ueber die stetige Abbildung einer Line auf ein Flächenstück.*Mathematische Annalen*, *38*(3), 459-460.

[11] Inoue, K., & Urahama, K. (2009). Halftoning with minimum spanning trees and its application to maze-like images. *Computers & Graphics*, *33*(5), 638-647.

[12] Lee, J. H., & Hsueh, Y. C. (1994). Texture classification method using multiple space filling curves. *Pattern Recognition Letters*, *15*(12), 1241-1244.

[13] Lin, T. H., & Shih Z. C. (2015). Content-aware bilateral texture filtering. Master thesis, National Chiao Tung University.

[14] Mandelbrot, B. B. (1983). *The fractal geometry of nature*, 173. Macmillan.

[15] Markuš, N., Fratarcangeli, M., Pandžić, I. S., & Ahlberg, J. (2015). Fast rendering of image mosaics and ASCII art. In *Computer Graphics Forum, 34*(6), 251-261.

[16] Ostromoukhov, V., & Hersch, R. D. (1995). Artistic screening. In*Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 219-228. ACM.

[17] Otsu, N. (1975). A threshold selection method from gray-level histograms. *Automatica*, *11*(285-296), 23-27.

[18] Pang, G., Zhu, M., & Zhou, P. (2015). Color transfer and image enhancement by using sorting pixels comparison. *Optik-International Journal for Light and Electron Optics*, *126*(23), 3510-3515.

[19] Peano, G. (1890). Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, *36*(1), 157-160.

[20] Santamaria, C., Bober, M., & Szajnowski, W. (2004). Texture analysis using level-crossing statistics. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference, 2*, 712-715. IEEE.

[21] Secord, A. J. (2002). *Random marks on paper: subtitle non-photorealistic rendering with small primitives*. Doctoral dissertation, University of British Columbia.

[22] Velho, L., & Gomes, J. D. M. (1991). Digital halftoning with space filling curves. *ACM SIGGRAPH Computer Graphics*, *25*(4), 81-90.

[23] Ventrella, J. (2012). Brainfilling Curves - A Fractal Bestiary. Eyebrain Books.

[24] Ventrella, J. (2015). Fractal curves. Software. Retrieved from http://www.fractalcurves.com/.

[25] Weird Science (2008). Fascinating fractals in nature. Retrieved from http://www.oddee.com/item_96529.aspx.

[26] Wyvill, B. (2015). Painting with flowsnakes. In *Proceedings of the workshop on Computational Aesthetics*, 171-182. Eurographics Association.

[27] Zang, Y., Huang, H., & Zhang, L. (2014). Efficient structure-aware image smoothingby local extrema on space-filling curve. *IEEE transactions on visualization and computer graphics*, *20*(9), 1253-1265.