國立交通大學

多媒體工程研究所

碩　士　論　文

由 3D 模型自動產生由格子組成之歐普藝術影像

An Automatic System for Squared-Based Op Art Rendering from 3D Models

研　究　生：曾梓瑄

指導教授：施仁忠　教授

　　　　　魏德樂　教授

中 華 民 國 一 百 零 三 年 八 月

由 3D 模型自動產生由格子組成之歐普藝術影像

An Automatic System for Square-based Op Art Rendering from 3D Models

研 究 生：曾梓瑄　　Student：Tzu-Hsuan Tseng

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

魏德樂　　Prof. Der-Lor Way

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 2014

Hsinchu, Taiwan, Republic of China

中華民國一百零三年八月

由 3D 模型自動產生由格子組成之歐普藝術影像

研究生: 曾梓瑄　　　　　　　　指導教授: 施仁忠 教授

魏德樂 教授

國立交通大學多媒體工程研究所

摘　　要

歐普藝術中歐普是「Optical」，是一種視覺效應的藝術。歐普藝術作品經常利用簡單的幾何形狀規律地排列造成空間的錯覺。然而要建立這種能產生錯視覺的歐普藝術作品並非相當容易。本篇論文探討的是由方格組成的歐普藝術作品，藉由三維模型提供的深度資訊判斷哪些方格應該產生形變，形變之後被放大的方格會有膨脹的效果，就像是在二維空間中凸出一顆球。此外，系統會對所有方格分別做形變，目的是希望做出的歐普藝術作品不僅有凸出來的效果，還能兼具描繪三維模型本身表面的凹凸結構。藉由使用我們的系統，使用者能夠輕鬆地繪出由方格組成的歐普藝術作品。

An Automatic System for Squared-Based Op Art Rendering from 3D Models

Student: Tzu-Hsuan Tseng Advisor: Prof. Zen-Chung Shih

Prof. Der-Lor Way

Institute of Multimedia Engineering

National Chiao-Tung University

ABSTRACT

Op Art, also called Optical Art, is a type of visual art that associated with optical illusions. Op art painters created optical spaces through the manipulation of simple repetitive forms. However, manual design of Op artwork is not an easy task. In this thesis, we focus on the Op artworks which are formed by squares. Instead of thinking which squares should be deformed, user can select a 3D model as input which already gives the depth information. By using the depth information, we can deform the squares to create 3D-like illusion which is like a ball. Besides, we propose a method to show the surface structure by perturbing the points by the neighbors of the square. Our system enables users to create the square-based Op artwork easily.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisors, Prof. Zen-Chung Shih and Prof. Der-Lor Way 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 classmate Jhen-Yao Hong who discussed with me when I was confused. I am also grateful to Chia-Min Hsu and Te-Mei Wang who gave me many suggestions on my thesis. Thanks for Wei-Shou Yang who have supported me during these months. Finally, I want to dedicate the achievement of this work to my family.

Contents

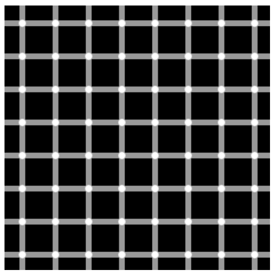
List of Figures

List of Tables

Chapter 1

Introduction

Op Art, also called Optical Art, is a type of abstract art that associated with optical illusions which deceive the human visual system into perceiving something that does not exit. The optical illusion is an interesting field. People love optical illusions and enjoy being deceived in a pleasant and surprising way. There are two main types of optical illusions. The first type is physiological illusion. This type of illusion is caused by a physiological imbalance that creates persistence of vision and alters the perception. For example, the Hermann grid illusion which is reported by Ludimar Hermann [8], as shown in Figure 1.1(a). The grey spots are perceived at the intersections of a white lines. But when we looking at the intersections directly, the grey spots disappear. The second type is the cognitive illusion. The psychologist called Gestalt believed that people organize some features to form a meaningful whole [21]. For example, one of the well-known ambiguous figures called the rabbit-duck illusion which a rabbit and a duck can been seen in a single image [7], as shown in Figure 1.1(b).


(a)

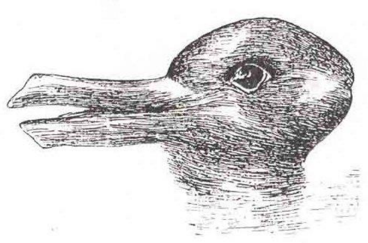

(b)

Figure 1.1: (a) The Hermann grid. (b) Rabbit-duck illusion.

We always comprehend whether a painting is considered good or bad by the professional critics. However, a layman can know the Op Art paintings without the background in art. Op Art is sudden and immediate to the layman [14]. Even a child of five said while viewing an exhibit of the Op Art, “The paintings are playing with me!” The first major international exhibition of op art was "The Responsive Eye" [18], held at the Museum of Modern Art, New York, in 1965. This exhibition was popular to the public. The principal artists of the Op movement as it emerged in the late 1950s and 1960s were Victor Vasarely, Bridget Riley, Richard Anuszkiewicz, Larry Poons, and Jeffrey Steele. Op art painters created optical spaces through the illusory manipulation of such simple repetitive forms as parallel lines or checkerboard patterns. Some may produce implied movement by varying the density of the patterns. Bridget Riley created vivid dynamic illusion in static pictures [16]. For example, a piece by Bridget Riley entitled *Movement in Squares*, as shown in Figure 1.2. The impression is given of warping by altering the size of some particular squares. Besides, Zanker et al. [22] analyzed the illusory motion in the Op artworks. Many researches may use multi-colored pattern to cause perceptual change [12], [11]. Still others focused on creating the illusory motion in a still image [4].

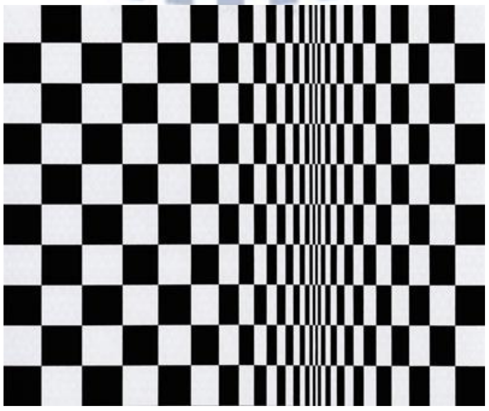


Figure 1.2: *Movement in Squares*, by Bridget Riley 1961.

Op Art works usually make use of simple geometric shapes such as cycles and line segments [13]. The use of pattern varies with each artist. Artists convey forms and shapes by packing these simple geometric shapes densely, as shown in Figure 1.3(a). Curves are also commonly used to create an illusion of depth, such as *Provocative Current* created by J*ulian Stanczak*, as shown in Figure 1.3(b), and *Zèbre* created by Vasarely, as shown in Figure 1.3(c). In addition, there are many of the well-known artworks which are only made in black and white, because highly contrasting colors make artworks appealing.

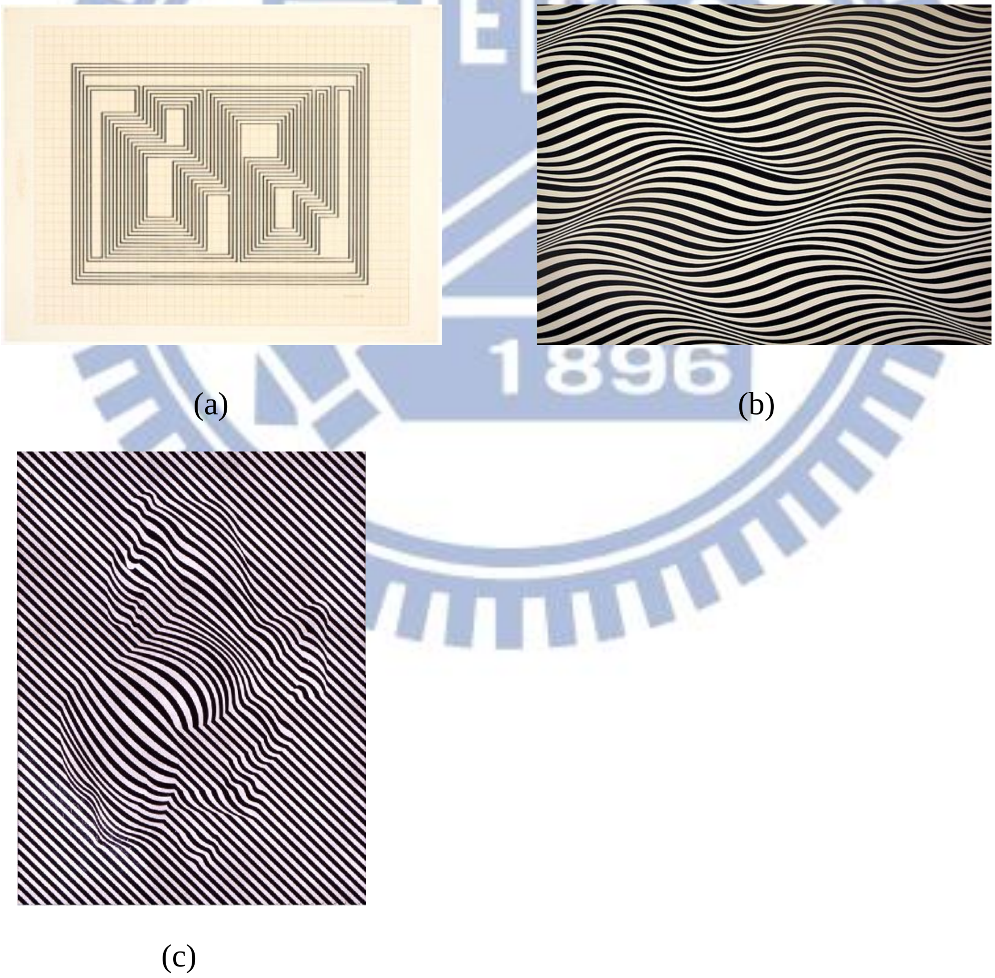


Figure 1.3: Example of Op Art formed by lines and curves: (a) Graphic Tectonic by Albers (1941). (b) Julian Stanczak's piece Provocative Current (1965). (c) *Zèbre* by

Vasarely (1944).

Some OP artists can create interesting 3D-like visual illusion such as impression of swelling or like a concave surface. A common technique in OP Art is the use of densely packed squares to depict some simple shapes. A piece by Vasarely entitled *Vega-Nor* comprises many squares to build a cycle, as shown in Figure 1.4. When the viewer looks at *Vega-Nor*, it is shaped something like a ball. This artwork captures the idea of creating 3D effect by altering the size or deforming the form of some particular squares.

On the other hand, the connections between the deformed squares and the original squares create the illusion of a boundary without actually drawing it. This phenomenon is known as an illusory contour or a subjective contour. In other words, illusory contours [15] are visual illusions that evoke the perception of an edge without a luminance or color change across that edge.

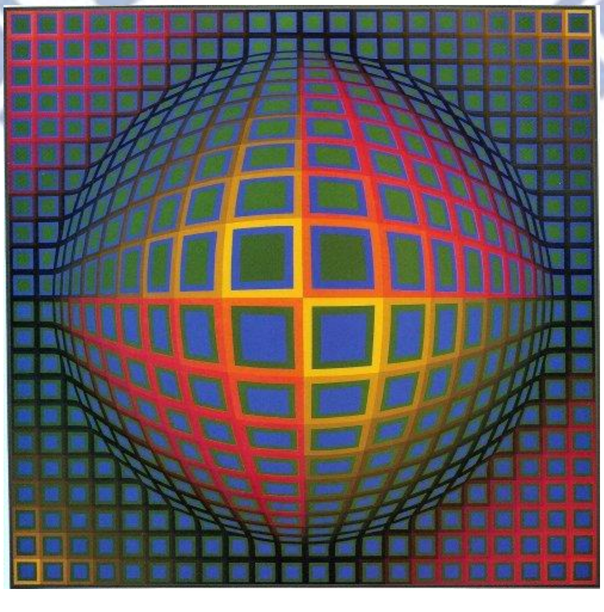


Figure 1.4: *Vega-Nor* by Vasarely (1969).

However, drawing such OP artworks formed with squares is not a trivial task especially for amateurs. Artists should know which squares need to be expanded and how large should these squares grow. In addition, the connections between the deformed squares and the original squares should be smooth and seamless. Thus, most OP artworks are handcrafted by skilled artists.

There has been some interest in Op Art in computer graphics and mathematics, such as the analysis of Bridget Riley's works. Dodgson [5] researched on the trade-off between regularity and randomness in Bridget Riley's early Op art. He considered how much randomness needs to be added to a regular pattern to create best aesthetic effect. Besides, Dunham [6] proposed a hyperbolic pattern of "squares" inspired by Vasarely's square grid patterns.

In this thesis, we inspire the square-based Op artworks, as shown in Figure 1.5. These works have the aspect of an elastic checkerboard in which inflations and shears have been applied. Instead of thinking which squares should be deformed, user can select a 3D model as input which already gives the depth information. If the squares have depth value, the squares are classed as object squares. Otherwise, the squares are classed as background squares. By using the depth information, we can deform the object squares to create 3D-like illusion and also deform some background squares for connecting smoothly. The impression is given of swelling on these expand squares.

The contributions of this thesis are as follows: (1) we present a clustering method to find the raised regions of the 3D model, so users do not need to choose the inflated squares manually; (2) a novel algorithm is applied to create the illusion which is like a bulge on the inflated squares; (3) we provide two rendered approaches to tint the Op Art results, so users can render the result in different styles.

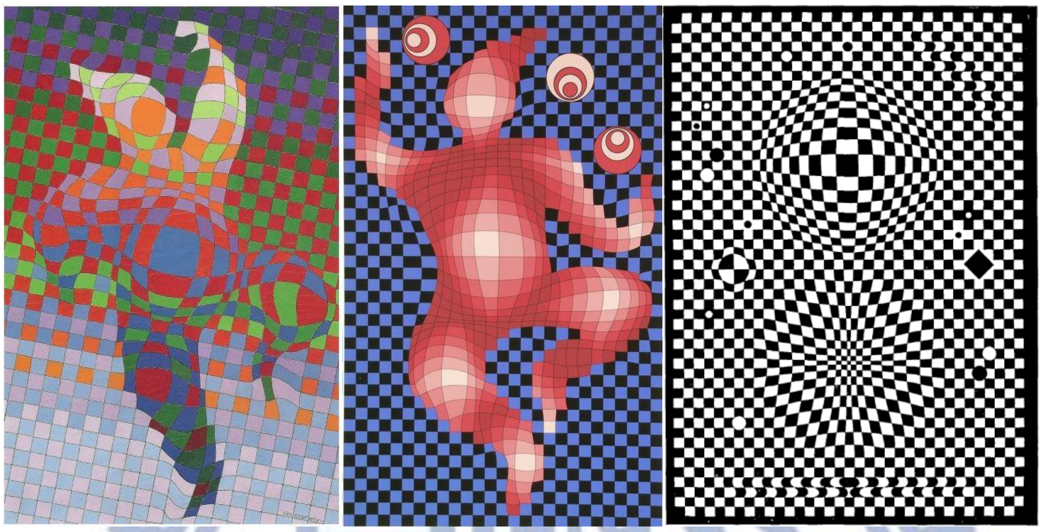


Figure 1.5: Examples of the square-based Op artworks. Vasarely's (a) *Harlequin,* (b) *The Juggler* and (c) *Vega-Os*.

The goal of this thesis is to create an automatic system for rendering Op artworks with squares. It is organized as follows: Chapter 2 reviews the related works about Op Art rendering with lines and curves, image warping using radial function and curvature-based drawing from 3D objects. Then, Chapter 3 describes the major method and our proposed algorithm that we used. Chapter 4 shows our experimental Op art results. Finally, the conclusion and future works are discussed in Chapter 6.

Chapter 2

Related Works

2.1 Op Art Rendering

One of the common ways in Op Art rendering is the use of line segments. If an artist wants to create a line-based Op artwork, the artifacts such as line breaks and T-junctions should be avoided. Inglis et al. [10] proposed a no artifacts method to create line-based Op art by taking an arbitrary 2-color or 3-color image as input.

The system is able to convey different color sections with parallel lines in different directions automatically. For example, take a 2-color image as input, as shown in Figure 2.1, the algorithm rasterizes the image to a square grid firstly. Then user can assign the line direction for each color region. Finally, the system replaces each region with parallel lines and the features of the input image are depicted without drawing the outlines. In other words, the line bends create illusory contour. Furthermore, transformations such as rotation and shearing can be applied to the lines to create a great diversity of outputs.

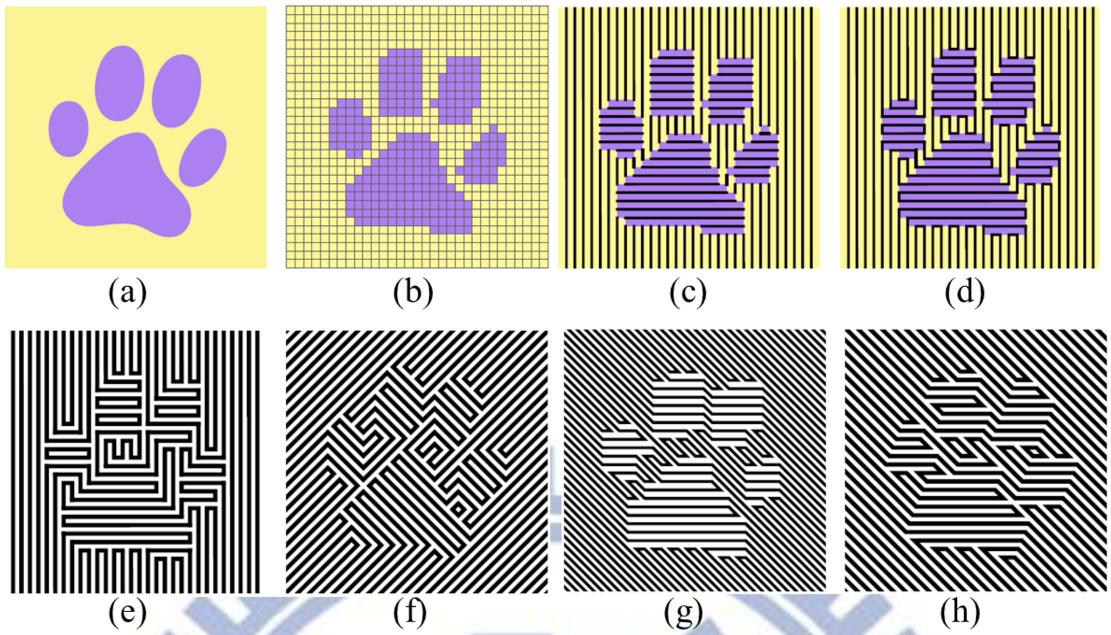


Figure 2.1: The 2-colour algorithm for creating line-based Op Art [10] is as follows: (a) input a 2-color image, (b) rasterize the image to a square grid, (c) fill each region with lines corresponding to the region color, (d) draw alternative edges along region boundaries, and (e) set the appropriate thickness of the lines. (f)(g)(h) transform the lines to get more different outputs.

Besides, Inglis et al. [9] also introduced an algorithm for creating curve-based Op Art. The steps for creating curve-based Op Art are as follows: to begin with, take an image or a 3D model as the input to compute the underlying vector field. Then curving the lines by the vector field and adjust the result to avoid too close neighboring paths. Finally, vary curve thickness to get a user preferences Op Art result, as shown in Figure 2.2.

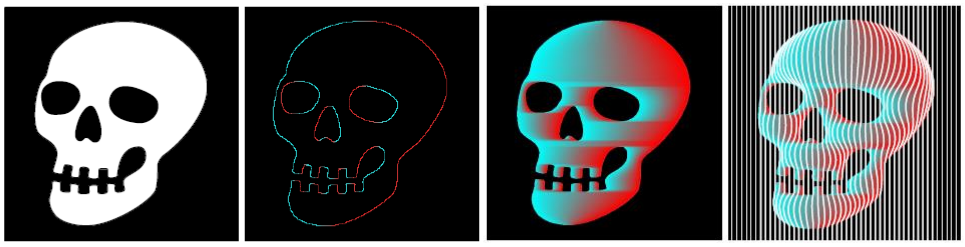


Figure 2.2: The illustrations for creating curve-based Op Art [9]. The steps of creating curve-based Op Art are as follows: (a) the input image, (b) apply Sobel filter to do edge detection, (c) compute the underlying vector field from the horizontal gradient values, (d) a set of vertical paths curved according to the vector field.

2.2 Image Warping by Radial Basis Function

A radial basis function (RBF) is a function whose value depends on the distance between some specific points to other points.

A warping of an image is a transformation to itself. One of the methods of image warping is using the RBF. Arad et al. [1] described an approach to the human facial impression warping by the RBF. The proposed approach makes use of a small amount of anchor points whose mapping coordinate is predetermined, as shown in Figure 2.3. The input points are anchor points $\mathrm{P} = \{p_1, p_2, \dots, p_n\}$ and $\mathrm{Q} = \{q_1, q_2, \dots, q_n\}$. Then we are looking for the warping function $\mathrm{T(p) = A(p) + R(p)}$ where the $\mathrm{A(p)}$ is the affine transformation and the $\mathrm{R(p)}$ is the radial basis function, so that the function can transform each p_i to q_i. Besides, other points also apply the same warping function to get a new coordinate.

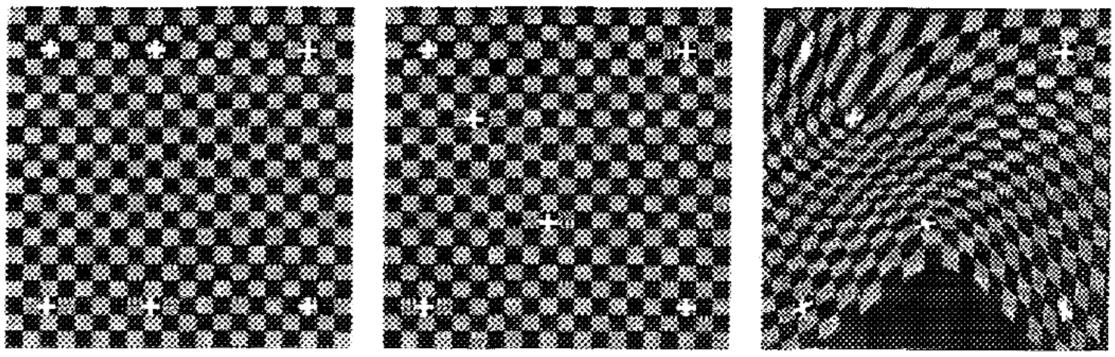
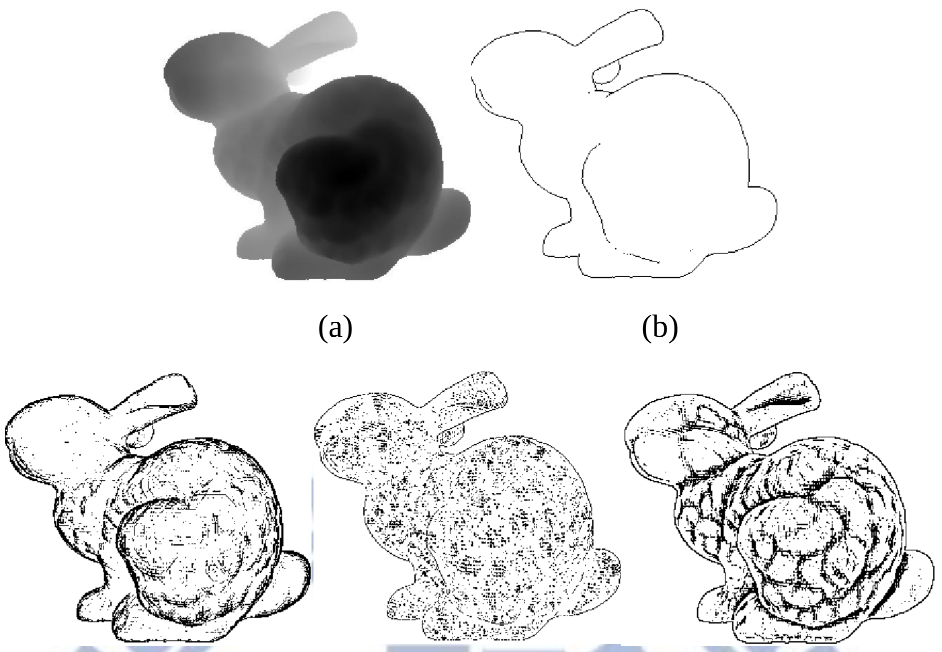


Figure 2.3: Example of image warping [1]: (a) source image with 6 specific anchor points marked by cross, (b) the destination of the 6 anchor points marked by cross, (c) the result of applying a radial basis function.

2.3 Curvature-based Drawing from 3-D Polygonal Objects

When we render a 3D model as a line drawing, we first draw the contours of the model. But the contours are quite limited in the information to convey about the shape. So we need to give more information such as depth discontinuities.

Chang [3] proposed a system to create curvature-based pen-and-ink drawings from 3D polygonal objects. According to the method, we first extract the depth information. Then calculate three different terms on the depth information. They are mean curvature, Gaussian curvature, and Laplacian convolution. The result is more perceptible and easier to infer the shape of object by viewer, as shown in Figure 2.4.


(a) (b)

(c) (d) (e)

Figure 2.4: Comparision of several visual effects. (a) depth map, (b) only contours, (c) result by mean curvature, (d) result by Gaussian curvature, (e) result by Laplacian convolution.

Chapter 3

Algorithm

3.1 System Overview

The goal of this thesis is to create Op-Art illusion by deforming or inflating some particular squares. Figure 3.1 describes an overview of our square-based Op Art system architecture. First, we construct a grid of squares by a user given 3D model using the user desired square number and square spacing. Then the rendered image is rasterized to the grid of squares. Each square has a corresponding depth value which is extracted from the depth buffer. In order to find the raised regions of the input 3D model, we use peak-climbing clustering algorithm [17] to cluster the squares by using the depth information.

After finding the center square each cluster, we use the cluster center as the kernel to shift the points of object squares. The displacements are according to the distance between points and the kernel by the radial basis function. Because of the line between background squares and object squares are not connected after points shift, we need to align the object squares with the background square smoothly and seamlessly.

In order to get a perceptible Op Art result, we make use of the depth value between two connected squares to perturb the in-between points, as shown in Figure 3.1(f). Finally, we obtain the square-based Op Art result by combining all the above steps.

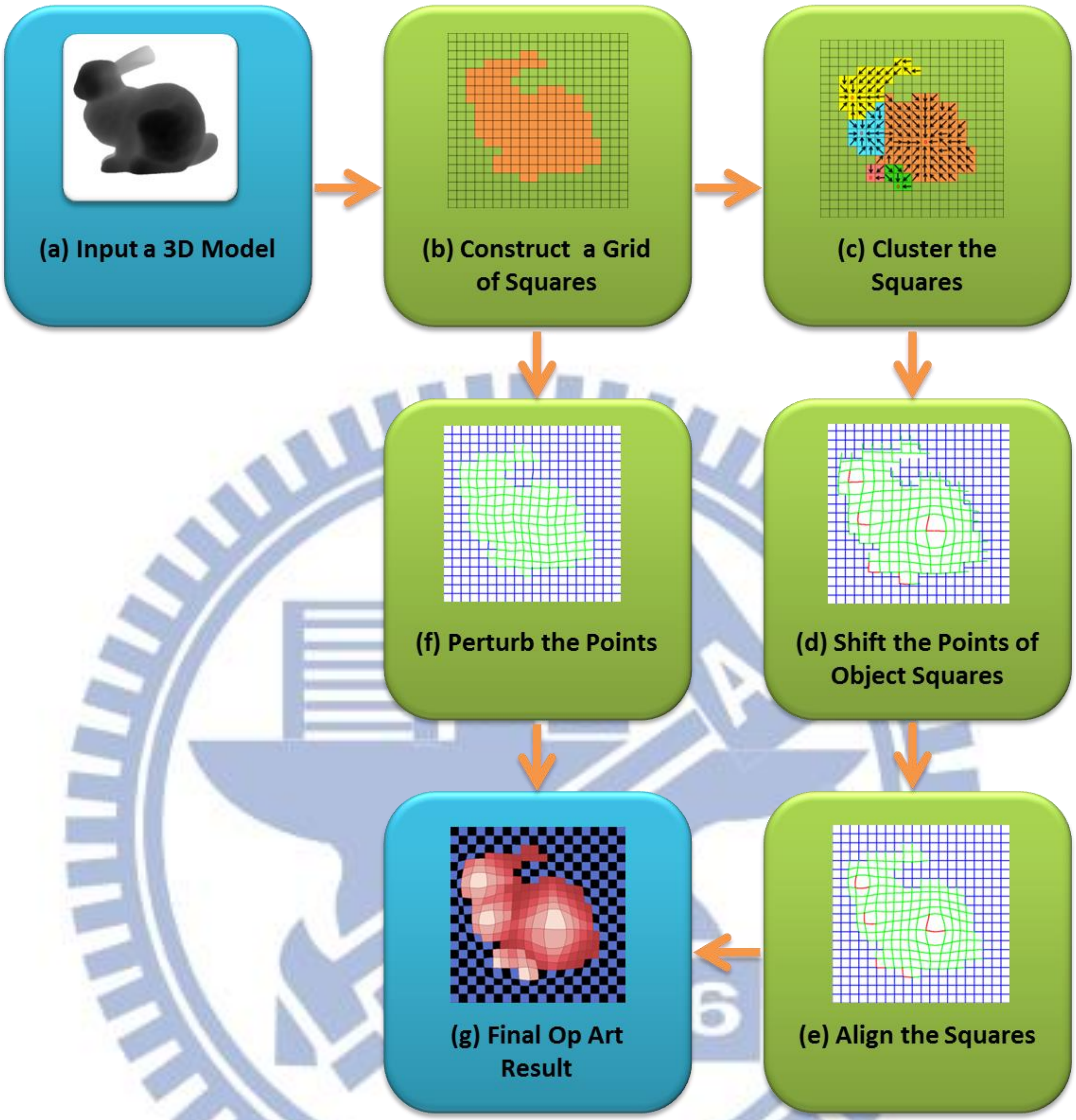


Figure 3.1: Overview of square-based Op Art system architecture. The following steps produce square-based Op Art form the 3D model.

3.2 Grid of Squares Construction

We construct the 3D polygonal object from the vertex, normal and face lists. After the image is rendered, the depth map is stored in the depth buffer. User specifies the input 3D model and the desired square spacing. Next, we construct a grid of squares, each size is $s \times s$. Then rendered image is rasterized to a square grid so that each square gets one color, as shown in Figure 3.2.

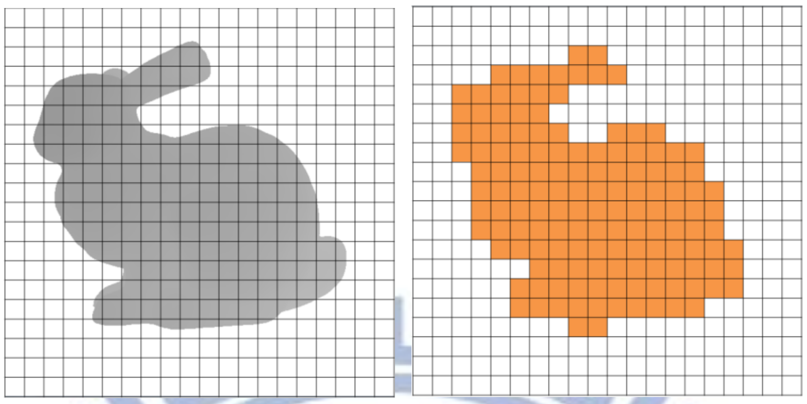

(a) (b)

Figure 3.2: (a) Rendered image of the input 3D model. (b) If the center pixel of square is covered by the rendered image, the square is marked as an object square. Otherwise, the square is a background square.

3.3 Squares Clustering

In order to reveal the 3D-like effect, we regard the raised regions as the regions to be inflated. Peak-climbing algorithm [17], also called hill-climbing search algorithm, is used to find the raised squares and clusters the object squares by using the depth values. We assign the depth value of each square as the depth of the square center pixel. For each object square, it looks at its eight neighbors to find the neighbor which has the smallest depth value. Then the square connects to its neighbor with smallest depth by an arrow, as shown in Figure 3.3(a).

Peak-climbing is a greedy local search because it grabs a good neighbor without thinking where to go next. But it is good for finding the local protruding squares. The cluster center is a “peak” where no neighbor has a lower depth value. In other words, the square receives arrows from all neighbors. Besides, we assign the object square to a cluster center by tracing the arrow directions until reaching a cluster center. So each object square belongs to one cluster center. The object squares which are assigned to the same center are in the same cluster. Therefore, we can get the cluster centers and its corresponding member squares by using peak-climbing algorithm as shown in Figure 3.3(b).

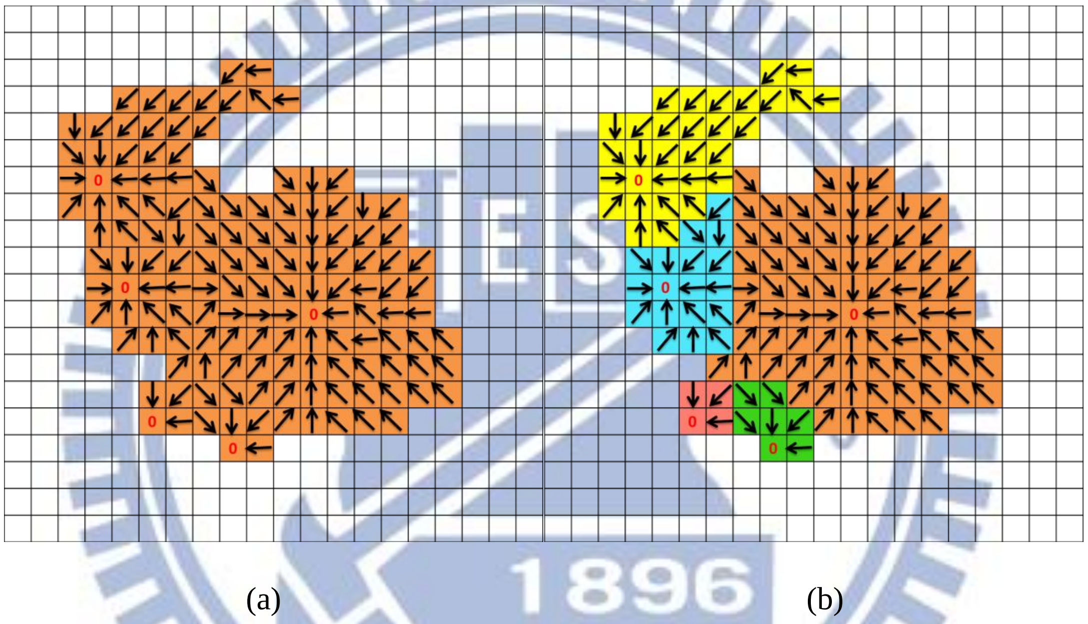


Figure 3.3: Clustering by peak-climbing algorithm: (a) The cluster center is marked as 0. There are five cluster centers in this example. (b) The different colors represent the different clusters.

3.4 Object Squares Deformation

Because the cluster centers are the raised regions, it is directly perceived that the closer the square to the cluster center, the more the square needs to be expanded. The goal of the algorithm is to create the shape which is like a raised ball in the cluster centers. We achieve the idea by making use of a simple radial basis function (RBF) to shift the points of the object squares by the distances between the points and the cluster centers.

Like many other rendering styles, it is difficult to select a set of parameters to achieve the viewer's desired effect. Our algorithm contains the following parameters:

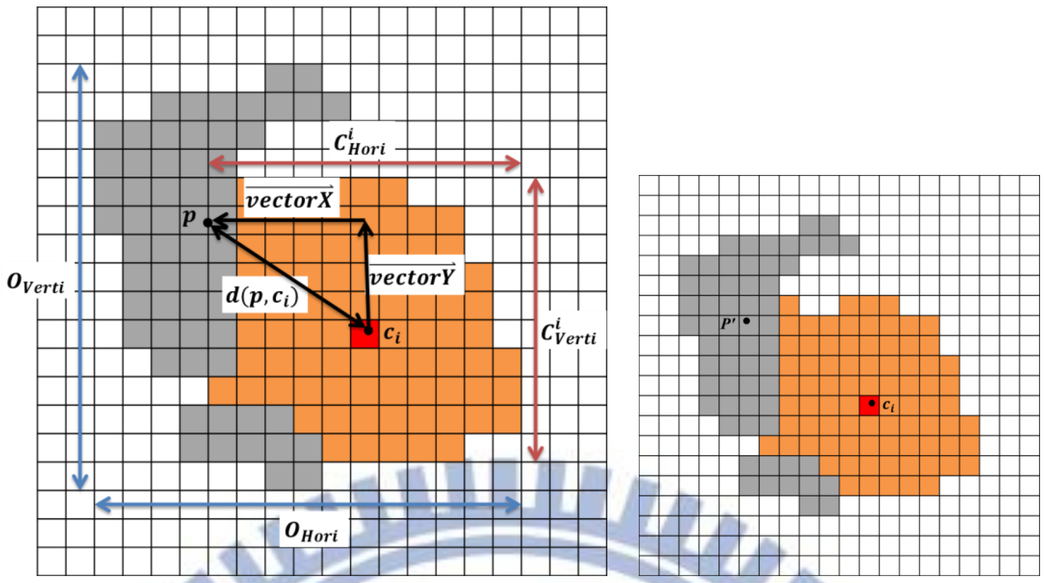
1. W and H are the width and height of the image, respectively;
2. N_{Hori} and N_{Verti} are the number of the horizontal squares per row and the number of the vertical squares per column, respectively;
3. $d(p, c_i)$ is the distance between the point p and the cluster center c_i;
4. d_{Min} and d_{Max} are the user specified lower bound and upper bound of $d(p, c_i)$, respectively;
5. S_{Min} and S_{Max} are the user specified lower bound and upper bound of the displacement of the points, respectively;
6. O_{Hori} and O_{Verti} are the number of the columns and the number of the rows covered by the object squares, respectively;
7. C^i_{Hori} and C^i_{Verti} are the number of the columns and the number of the rows covered by the squares belong to the cluster i, respectively;

Algorithm 1: Shift the Points of Object Squares

```
Input: $d_{Min}, d_{Max}, S_{Min}, S_{Max}$
Output: new coordinate $P'(x', y')$
1.  for all cluster center $c_i$ do
2.      $scale_X \leftarrow C^i_{Hori} / O_{Hori}$
3.      $scale_Y \leftarrow C^i_{Verti} / O_{Verti}$
4.      if $^{scale_X}/_{scale_Y} > k \ or < \frac{1}{k}$ then
5.          $scale_X, scale_Y \leftarrow \min(scale_X, scale_Y)$
6.      for all points $p(x, y) \in$ object squares do
7.          compute $\overrightarrow{vectorX}, \overrightarrow{vectorY}$ and $d(p, c_i)$
8.          $d_{nor}(p, c_i) \leftarrow$ normalize $d(p, c_i)$ to $[d_{Min}, d_{Max}]$
9.          $m_x \leftarrow$ normalize $\frac{\overrightarrow{vectorX}}{d_{nor}(p,c_i)^2}$ to $[S_{Min}, S_{Max} \times scale_X]$
10.         $m_y \leftarrow$ normalize $\frac{\overrightarrow{vectorY}}{d_{nor}(p,c_i)^2}$ to $[S_{Min}, S_{Max} \times scale_Y]$
11.         $x' \leftarrow x + m_x$
12.         $y' \leftarrow y + m_y$
```

We propose an algorithm which shifts the points of object squares in Algorithm 1. On the one hand, the user should input d_{Min} and d_{Max} for determining the influence on $d(p, c_i)$. When d_{Min} is fixed, the larger d_{Max} represents that the displacements of the points will more sensitive to $d(p, c_i)$. On the other hand, $S_{Min}$and S_{Max} should be given to decide how large the squares will be inflated.

For each cluster center c_i, we determine how large the squares should be inflated by the number of the rows and the columns covered by the cluster. If the number of the rows and the columns covered by the cluster are the largest, the squares in this cluster will be inflated more than those in other clusters. We calculate C^i_{Hori} and C^i_{Verti} as the number of columns and the number of rows covered by the cluster c_i, as shown in Figure 3.4(a). Then we calculate $scale_x$ and $scale_y$to scale the maximum displacement S_{Max}. To avoid the squares do not inflate like a ball, we set $scale_x$ and $scale_y$ to the same value when these two value are quite different.



(a) (b)

Figure 3.4: Example of a point shifts according to $\overrightarrow{vectorX}$, $\overrightarrow{vectorY}$ and $d(p, c_i)$. (a) The cluster center c_i is used to shift all the points of object squares. (b) The new coordinate P' shifted by the cluster center c_i.

We shift the points which belong to the object squares by the cluster centers, as shown in Figure 3.4. After the points shift according to Algorithm 1, the preliminary result has been made, as shown in Figure 3.5. But the result contains some dislike appearances such as the overlapped squares on the boundary between the object squares and the background squares, and the holes between the connected points. We will solve these undesired appearances in the next section.

Figure 3.5 displays the result of shifting the points by our Algorithm 1.The blue squares are the background squares which are motionless as the rasterized square grid. The green points and red points belong to the object squares. These points are shifted by the cluster centers marked by red. In this example, the parameter are set as follows: $W = H = 600$, $N_{Hori} = N_{Verti} = 20$, $d_{Min} = 1$, $d_{Max} = 13$, $S_{Min} = 0$, $S_{Max} =$

30.

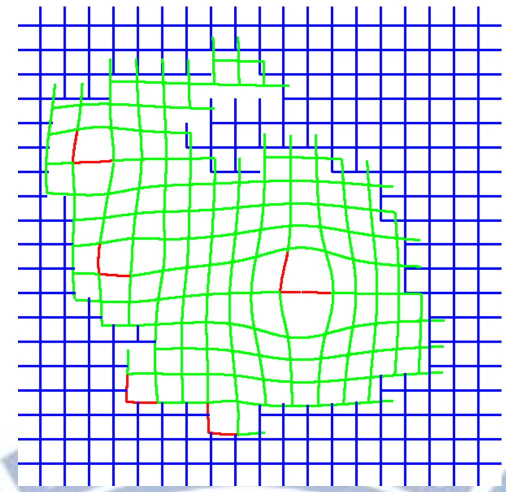
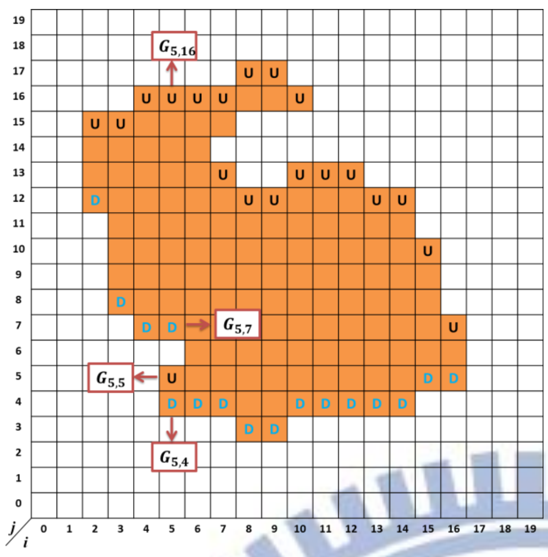


Figure 3.5: The result of shifting the points by our Algorithm 1.

3.5 Squares Alignment

To avoid the object and background squares being overlapped, we need to align them flawlessly. Firstly, we find the up, down, left and right borders of the object squares. Define $G_{i,j}$ be the square on column i and row j. If $G_{m,n}$ be an object square and $G_{m,n+1}$ is a background square, then $G_{m,n}$ belongs to the up border. Likewise, if $G_{m,n}$ be an object square and $G_{m-1,n}$ is a background square, then $G_{m,n}$ belongs to the left border. The down borders and the right borders are found in the same way.

Figure 3.6(a) shows the up borders and the down borders which are marked as **U** and **D**, respectively; Figure 3.6(b) illustrates the left borders and the right borders marked as **L** and **R**, respectively. There are two squares belong to the up border in column 5, $G_{5,5}$ and $G_{5,16}$. $G_{5,16}$ is the top most up border among them. There are two squares belong to the down border in column 5, $G_{5,4}$ and $G_{5,7}$. $G_{5,4}$ is the bottom down border among them, as shown in Figure 3.6.


(a)

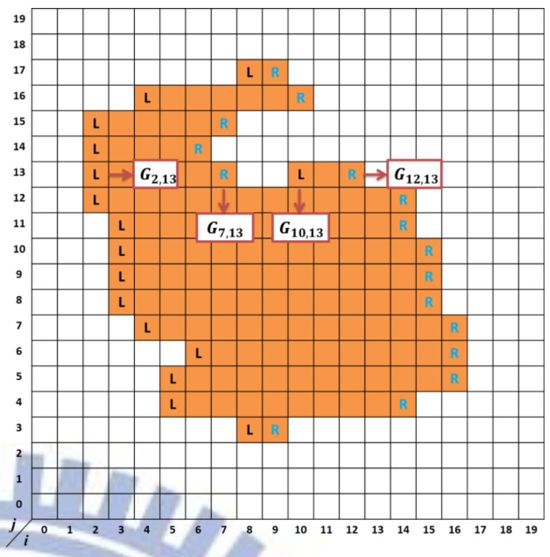

(b)

Figure 3.6: The borders of the object squares.

Then we calculate the displacement after the point shifted on each border. Finally, add the negative displacement to the points of object squares. We obtain a result that the edges of the quadrilateral are connected seamlessly, as shown in Figure 3.7.

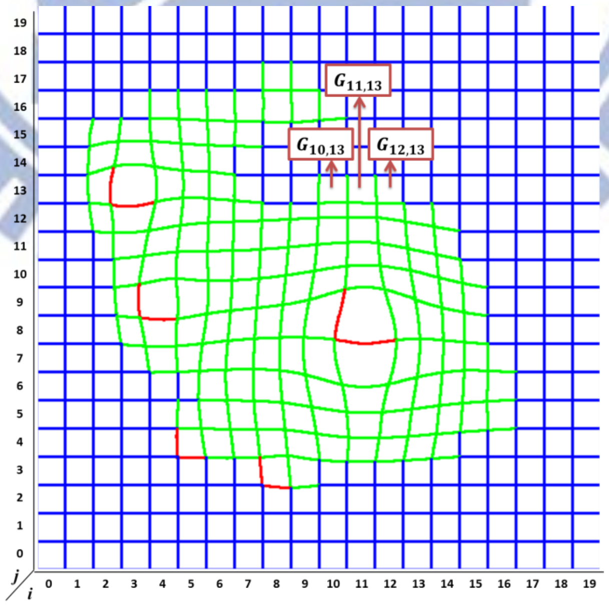


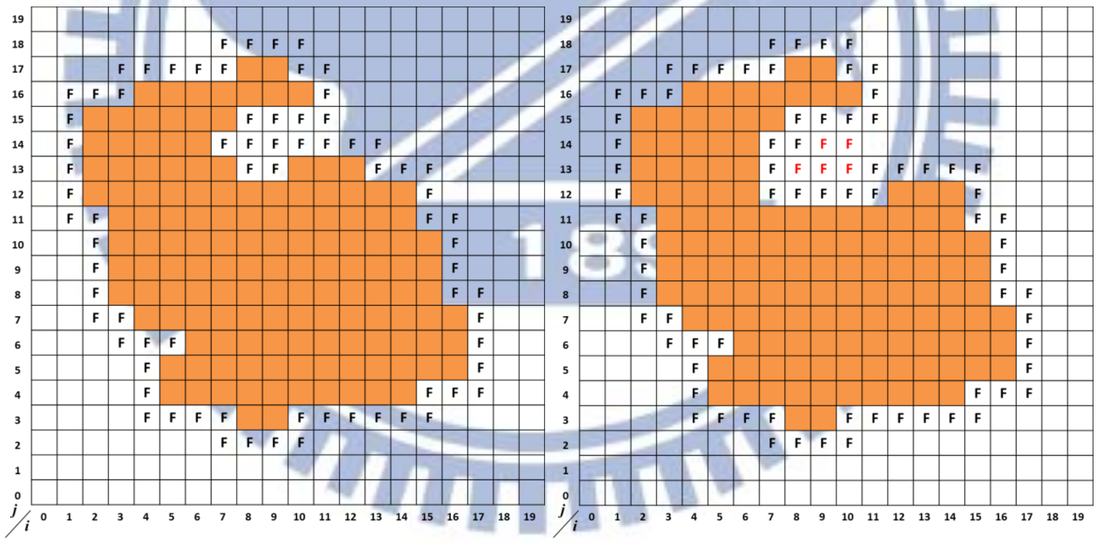
Figure 3.7: The result after deal with the boundaries alignment. The object squares and the background squares connect seamlessly.

The result is still unnatural because all the background squares are static. After we aligned all the border squares to the background squares, some squares will look strange. For example, the squares on the back of bunny are odd because these squares are not like a part of the bunny. On the contrary, these squares look like a part of background, as $G_{10,13}$, $G_{11,13}$ and $G_{12,13}$ in Figure 3.7. In order to solve this unnatural appearance, our system creates a type of square called fake object square. In fact, the fake object squares are the background squares. But we regard the fake object squares as the object squares and shift the points in the fake object squares as we do in the object squares.

We use Algorithm 2 to find the fake object squares. For Figure 3.8(a), we can find all the fake object squares simply by step 2 to step 5 of Algorithm 2. But for Figure 3.8(b), the fake object squares in red color need to be found by step 6 to step 17 of Algorithm 2. Figure 3.9(a) and 3.9(b) are the results with and without step 6 to 17. The red colored squares in Figure 3.8(b) will be deformed by Algorithm 1 and the result looks more natural, as shown in Figure 3.9(a). On the contrary, these squares in Figure 3.9(b) are treated as background, so the squares around these background squares are almost static since they need to be aligned, as shown in Figure 3.9(b).

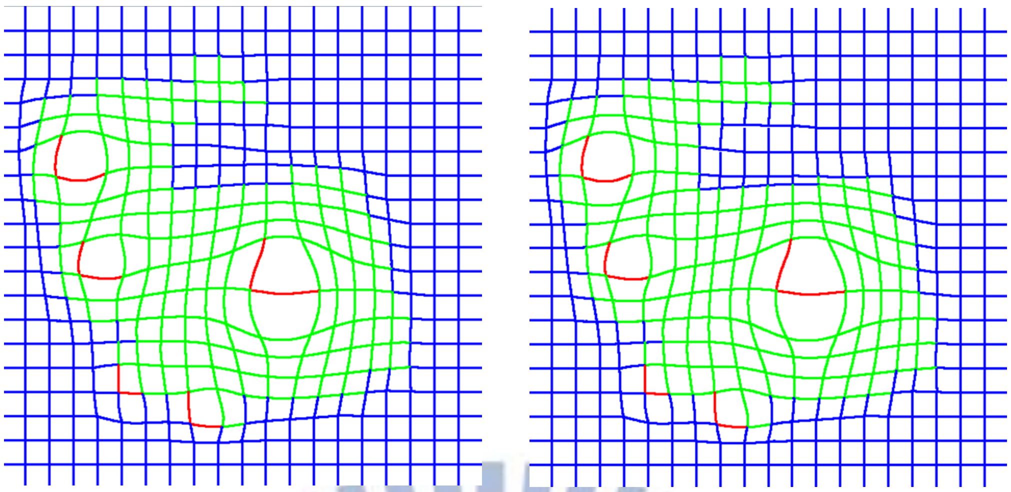
Algorithm 2: Find the Fake Object Squares

```
1.  **Output:** the fake object squares
2.  **for all** $G_{i,j}$ ∈ border **do**
3.      **for** $G_{m,n} = G_{i-1,j-1}, G_{i,j-1}, G_{i+1,j-1}, G_{i-1,j}, G_{i+1,j}, G_{i-1,j+1}, G_{i,j+1}, G_{i+1,j+1}$
4.          **if** $G_{m,n}$ ∈ background square **then**
5.              $G_{m,n}$ ∈ fake object square
6.  **for all** $column_i$ **do**
7.      $G_{i,j\_up}$ ← the highest up border in $column_i$
8.      $G_{i,j\_down}$the lowest down border in $column_i$
9.      **for** $k = j\_down \dots j\_up$ **do**
10.         **if** $G_{i,k}$ ∈ background square **then**
11.             $G_{i,k}$ ∈ fake object square
12. **for all** $row_j$ **do**
13.     $G_{i\_left,j}$ ← the left most left border in $row_j$
14.     $G_{i\_right,j}$← the right most right border in $row_j$
15.     **for** k = $i\_left \dots i\_right$ **do**
16.         **if** $G_{k,j}$ ∈ background square **then**
17.             $G_{k,j}$ ∈ fake object square
```



(a) (b)

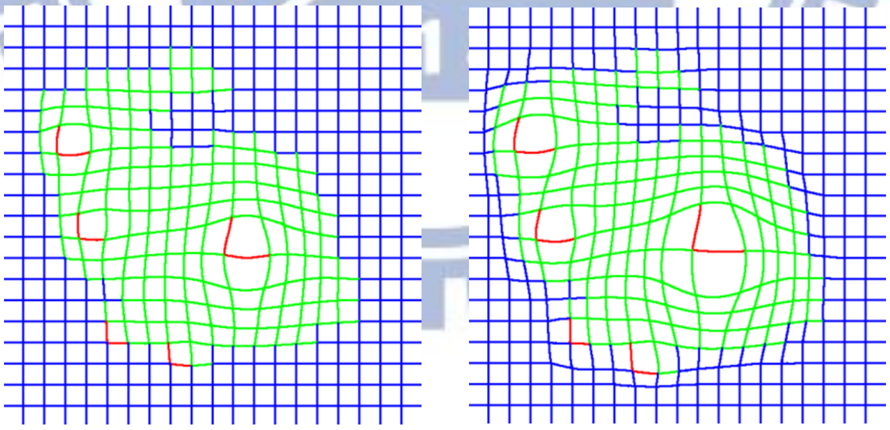
Figure 3.8: The fake object squares are marked as F.



(a) (b)

Figure 3.9: The results with and without step 6 to 17 of Algorithm 2.

We regard the fake object squares as object squares and shift the points using Algorithm 1. The result is more stereoscopic because the inflated effect decreases gradually from the object squares to the fake object squares, as shown in Figure 3.10(b).



(a) (b)

Figure 3.10: The compare of using the fake object squares or not. (a) The result without adding the fake object squares. (b) The result with the fake object squares.

3.6 Points Disturbance

The current result is only affected by the cluster centers. The inflated squares can be understood as the raised regions of the input 3D model. But the result does not account for the surface structure of the model. Besides, the squares farther from the cluster centers are almost static, such as the ears and the tail of the bunny, as shown in Figure 3.10(b). Therefore, we propose a method to show the surface structure by perturbing the points by the neighbors of the square, as shown in Algorithm 3. For each column, if the depth value of the square $G_{i,j}$ is bigger than the depth value of the square $G_{i,j-1}$, we perturb the points in this column to the right smoothly. On the contrary, we perturb the points in this column to the left smoothly. Likewise, we perturb the points in each row by comparing the depth values of two adjacency squares. By perturbing the points can we perceive the surface structure. Besides, we can observe that the squares deform obviously even when they are farther from the cluster centers, as shown in Figure 3.11.

Algorithm 3: Perturb the Points

```
1.  Output: new coordinate of the points
2.  for all $column_i$ do
3.      $C_i^{concave} \leftarrow 0$, $C_i^{canvex} \leftarrow 0$
4.      for $k = 1 \dots N_{Verti} - 1$ do
5.          if the depth of $G_{i,k-1}$ $<$ the depth of $G_{i,k}$ then
6.              $C_i^{concave} \leftarrow C_i^{concave} + 1$
7.          else then
8.              $C_i^{canvex} \leftarrow C_i^{canvex} + 1$
9.  for all $column_i$ do
10.     if $\|C_i^{concave} - C_i^{canvex}\| \leq \frac{N_{Verti}}{2}$ and $C_i^{concave} \neq 0$ and $C_i^{canvex} \neq 0$ do
11.         for $k = 1 \dots N_{Verti} - 1$ do
12.             if the depth of $G_{i,k-1}$ $<$ the depth of $G_{i,k}$ then
13.                 for $m = k \dots N_{Verti}$ do
14.                     shift the points of $G_{i,m}$ to right smoothly
15.             else then
16.                 for $m = k \dots N_{Verti}$ do
17.                     shift the points of $G_{i,m}$ to left smoothly
18. for all $row_j$ do
19.     $R_j^{concave} \leftarrow 0$, $R_j^{canvex} \leftarrow 0$
20.     for $k = 1 \dots N_{Hori} - 1$ do
21.         if the depth of $G_{k-1,j}$ $<$ the depth of $G_{k,j}$ then
22.             $R_j^{concave} \leftarrow R_j^{concave} + 1$
23.         else then
24.             $R_j^{canvex} \leftarrow R_j^{canvex} + 1$
25. for all $row_j$ do
26.     if $\|R_j^{concave} - R_j^{canvex}\| \leq \frac{N_{Hori}}{2}$ and $R_j^{concave} \neq 0$ and $R_j^{canvex} \neq 0$ do
27.         for $k = 1 \dots N_{Hori} - 1$ do
28.             if the depth of $G_{k-1,j}$ $>$ the depth of $G_{k,j}$ then
29.                 for $m = k \dots N_{Hori}$ do
30.                     shift the points of $G_{k,j}$ to down smoothly
31.             else then
32.                 for $m = k \dots N_{Hori}$ do
33.                     shift the points of $G_{k,j}$ to up smoothly
```

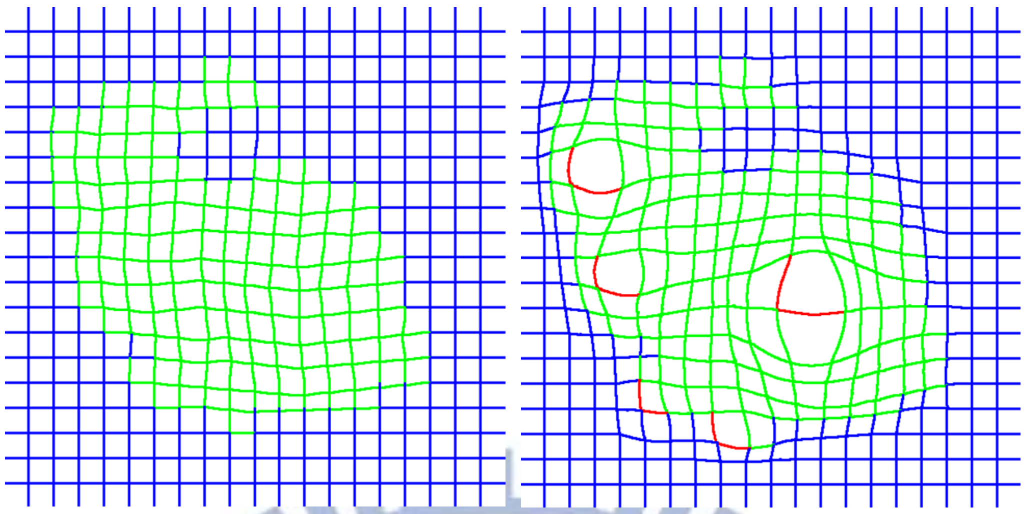
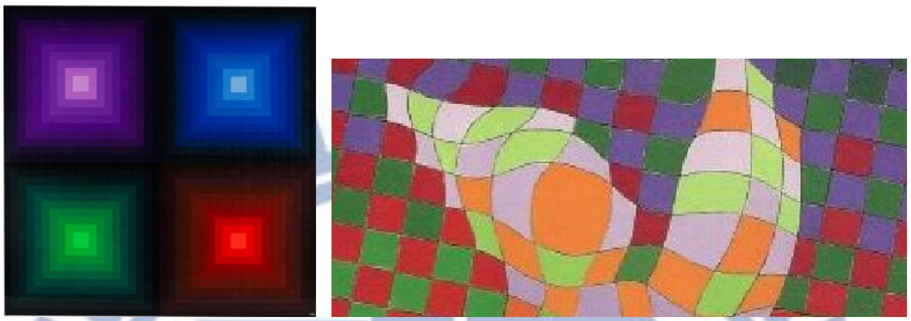


Figure 3.11: The result uses Algorithm 3 to perturb the points. (a) Only use Algorithm 3 to perturb the points. (b) Combine the result using Algorithm 1 and Algorithm 3.

3.7 Rendering Approaches to Op Art Results

After observing a great quality of square-based Op Art works drawn by artists, we select two rendering approaches to render our experiment results. The first type refers to the Vasarely illusion. This illusion is the perception of an oblique light cross which is like a ‘X” along the diagonals of concentric squares [2]. But this cross does not really exist. It has been incorporated into many Op Art paintings, as shown in Figure 3.12(a). Several research studies discussed the Vasarely illusion such as Tsofe et al. [20] and Troncoso et al [19]. The artworks of this type render the object squares by a luminance gradient, as shown in Table 3.1. The squares $G_{i,j}$ which is a cluster center is tinted by the color of index 1. Other squares $G_{m,n}$ are tinted by the color of index $\|m-i\|+\|n-j\|$. If the index is greater than the greatest index 8, we use index 8 as the index. We can create more colors for obvious perception of layers by interpolating the colors in Table 3.1. The second type of our rendering approach is inspired by the Vasarely's *Harlequin*. We stored the colors in a table. Then apply these colors to our results. For each row in *Harlequin*, the squares are tinted by four colors, two of them are for the object squares and the others are for the background squares, as shown in Figure 3.12(b).



(a) (b)

Figure 3.12: (a) The Op Art work *Arcturus*, by Vasarely. (b) A part of *Harlequin*, by Vasarely.

Table 3.1: The colors of a luminance gradient are rendered in our experiment results.

Index	1	2	3	4	5	6	7	8
color	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

Chapter 4

Experimental Results

We present the implementations and results in this chapter. The input sources are 3D triangle meshes, and the outputs are 2D square-based Op Art images. All the 3D models we used can be found from Google 3D Warehouse. These models are post-processed to show their depth maps, as shown in Figure 4.1. Even a skilled Op artist may spend several days drawing a painting. By using our rendering system, each result would be generated only in a few seconds.

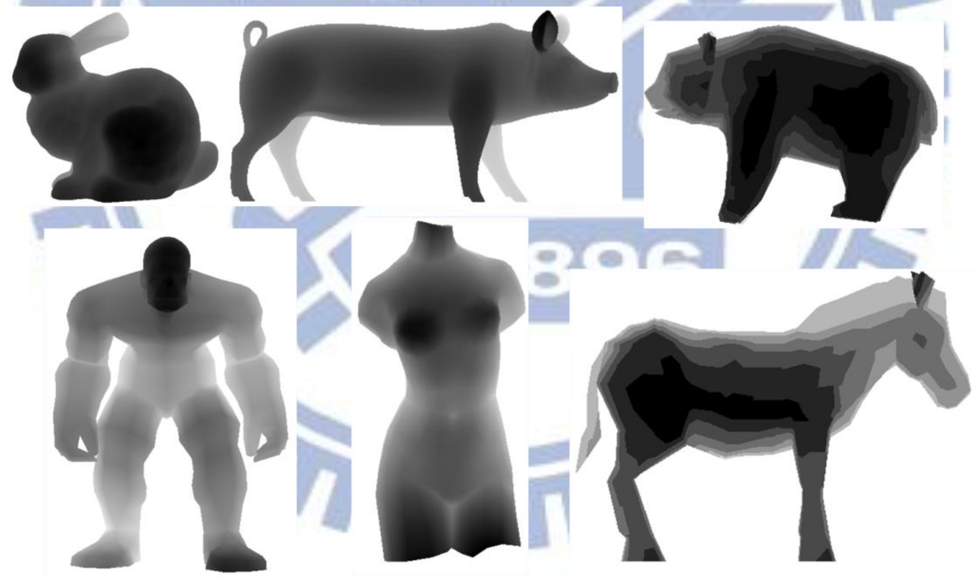


Figure 4.1: Depth maps of our 3D models.

In our system, users need to define some parameters for generating the desired Op Art images. Figure 4.3 compares four experimental results with different parameters S_{Max} and d_{Max}. The corresponding values of each result were defined in Table 4.1. Figure 4.3(a) was produced by the greatest value of S_{Max}. So the inflated effect of squares around each cluster center is more obvious than those in Figure 4.3(b)-(d). Figure 4.3(b) and Figure 4.3(c) used the same S_{Max}. Figure 4.3(c) looks smoother, especially the squares near to the cluster centers because the parameter d_{Max} is smaller.

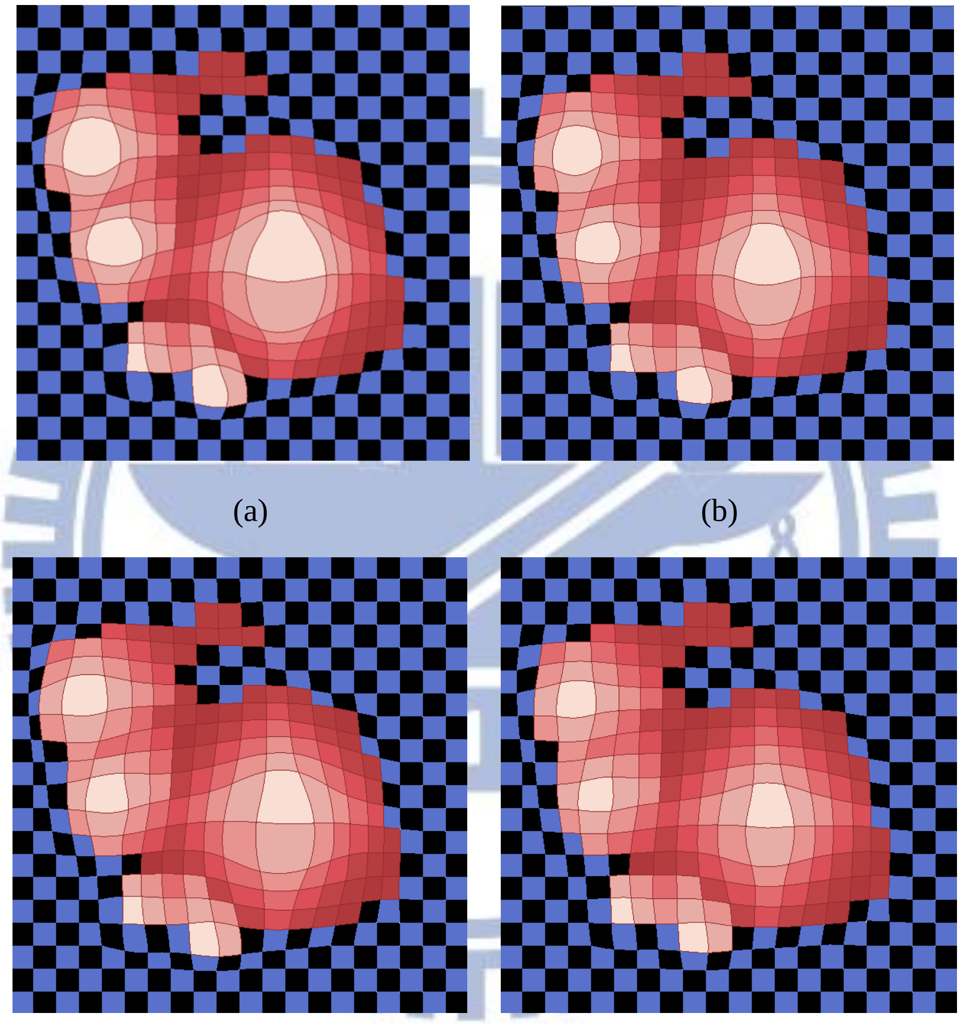


Figure 4.3: Op Art results with different values of parameters.

Table 4.1: The parameters of each experiment results. ($\boldsymbol{d_{Min} = 1}$, $\boldsymbol{S_{Min} = 0}$.)

Figure	S_{Max}	d_{Max}
Figure 4.3(a)	$\frac{W}{N_{Hori}} \times 2.0$	25
Figure 4.3(b)	$\frac{W}{N_{Hori}} \times 1.5$	20
Figure 4.3(c)	$\frac{W}{N_{Hori}} \times 1.5$	14
Figure 4.3(d)	$\frac{W}{N_{Hori}} \times 1.0$	14

Figure 4.4 shows the result of the pig model. We first show the depth map of the model and its corresponding clustering results, the different clusters are represented by different colors, as shown in Figure 4.4(b). Then we render the object squares of the pig model by a luminance gradient which creates the Vasarely illusion. The parameter N_{Hori} and N_{Verti} of Figure 4.4(d) are greater than those in Figure 4.4(c). When the number of square increases, there is a seeming lightening at the edges, forming the "X" [2]. We perceive that the Vasarely illusion is clearer with larger N_{Hori} and N_{Verti}.

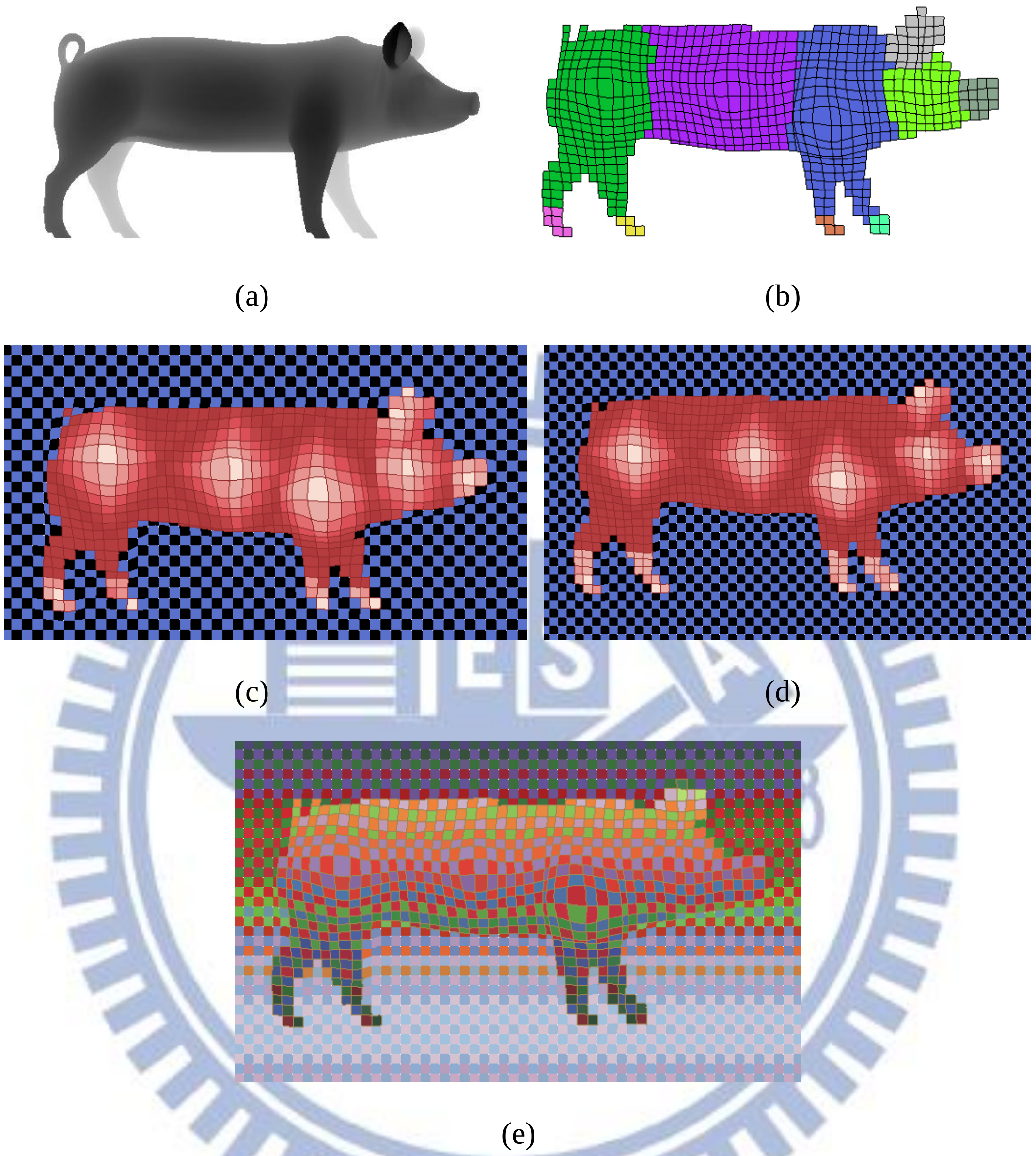
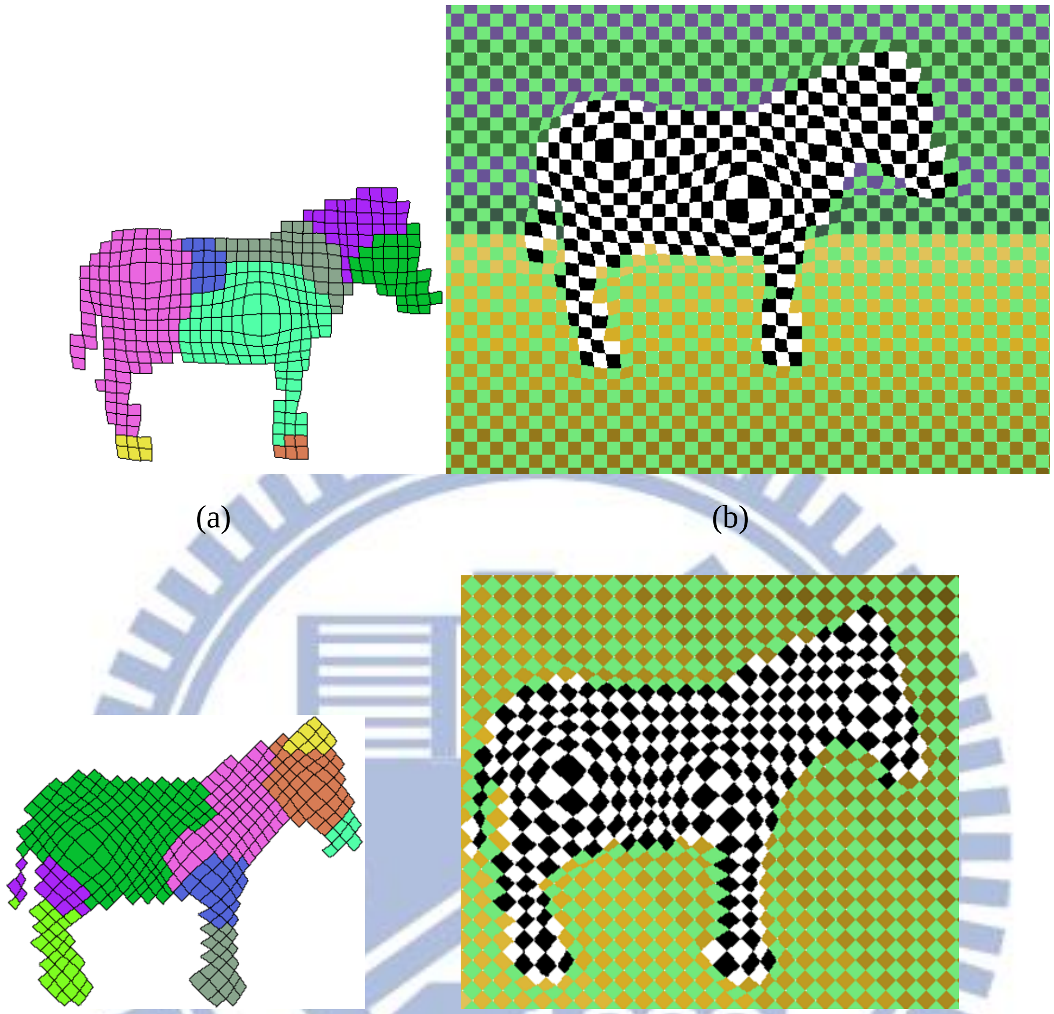


Figure 4.4: The rendered result of pig. (a) The depth map. (b) The clustering result. (c)-(e) The rendered Op Art images.

Figure 4.5 shows the clustering and rendering result of the zebra model. This model has eight clusters, as shown in Figure 4.5(a). We used black and white to tint the zebra because Op Art often uses two highly contrasting colors to render the artworks. Besides, we rotated the squares to make it diagonal, as shown in Figure 4.5(d).



(a) (b)

(c) (d)

Figure 4.5: The rendered results of zebra. (a)(c) The clustering results of the rendered Op Art images (b) and (d), respectively.

Figure 4.6 compares the result with and without using the fake object squares. The background squares close to Venus's loin are regarded as the fake object squares. So they are deformed along with the object squares, as shown in Figure 4.6(b). Besides, we compare the results with perturbing the points or not. In Figure 4.6(b), we perturb the points so that the thighs are more perceptible. On the contrary, the result which only used the cluster centers to shift the points is shown in Figure 4.6(c).

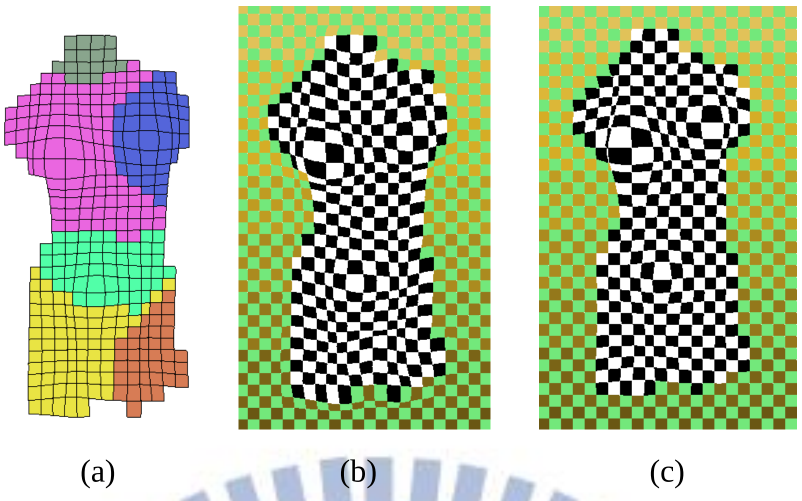


Figure 4.6: The rendered results of Venus. (a) The clustering results. (b)(c) The rendered Op Art images.

Finally, we compare our result with the Op Art painting *The Juggler* by Vasarely, as shown in Figure 4.7. The inflated effects on the raised regions look similar. Besides, the background squares left to the loin were deformed analogously. More Op Art experiment results created by our system are shown in Figure 4.8 to Figure 4.10.

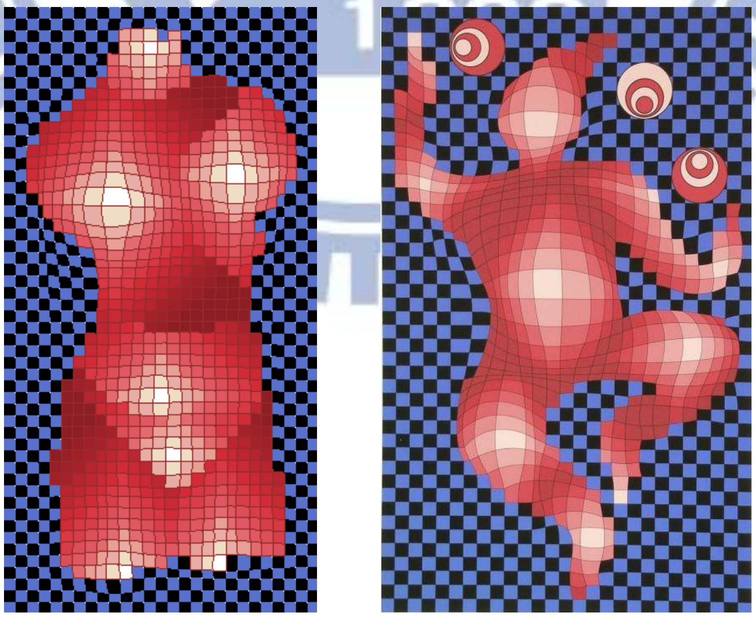


Figure 4.7: Compare our result with *The Juggler* by Vasarely. The left is the rendered result of Venus. The right is Vasarely's *The Juggler.*

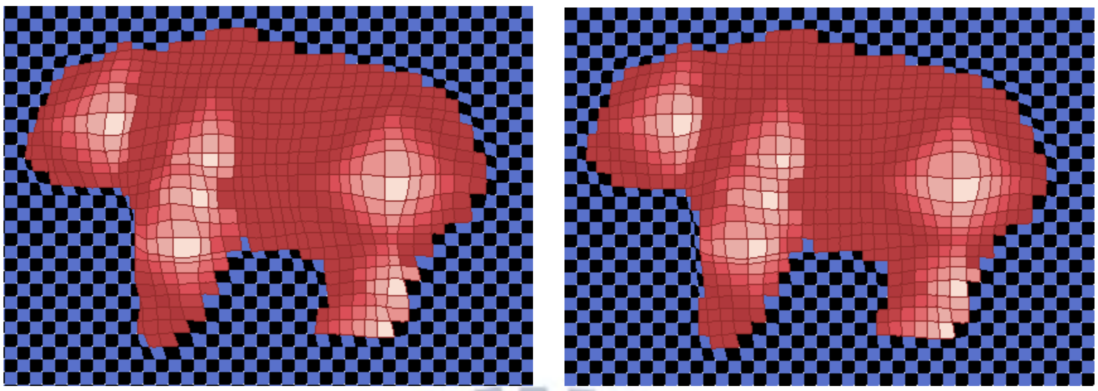


Figure 4.8: The rendered bear with perturbing the points or not.

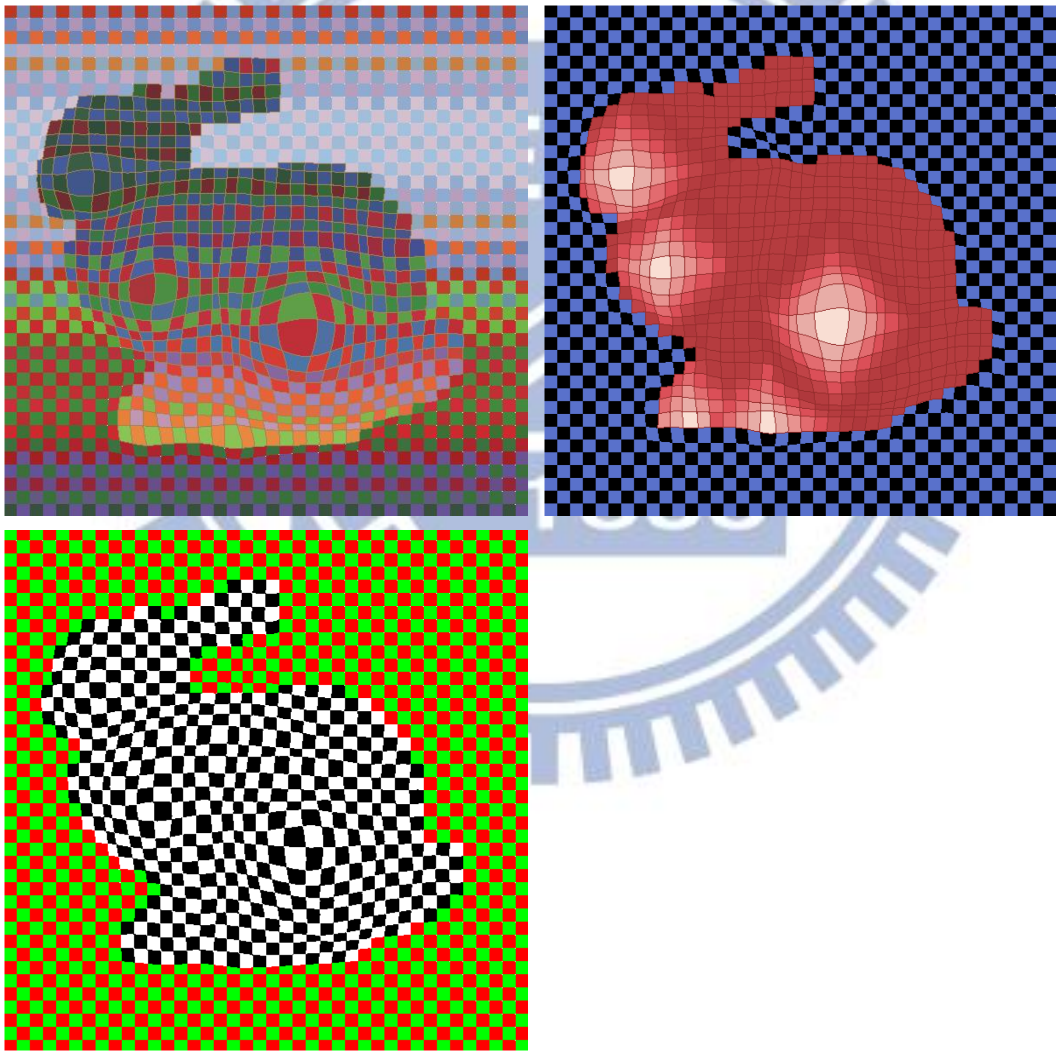


Figure 4.9: The Op Art results rendered by different approaches of the bunny.

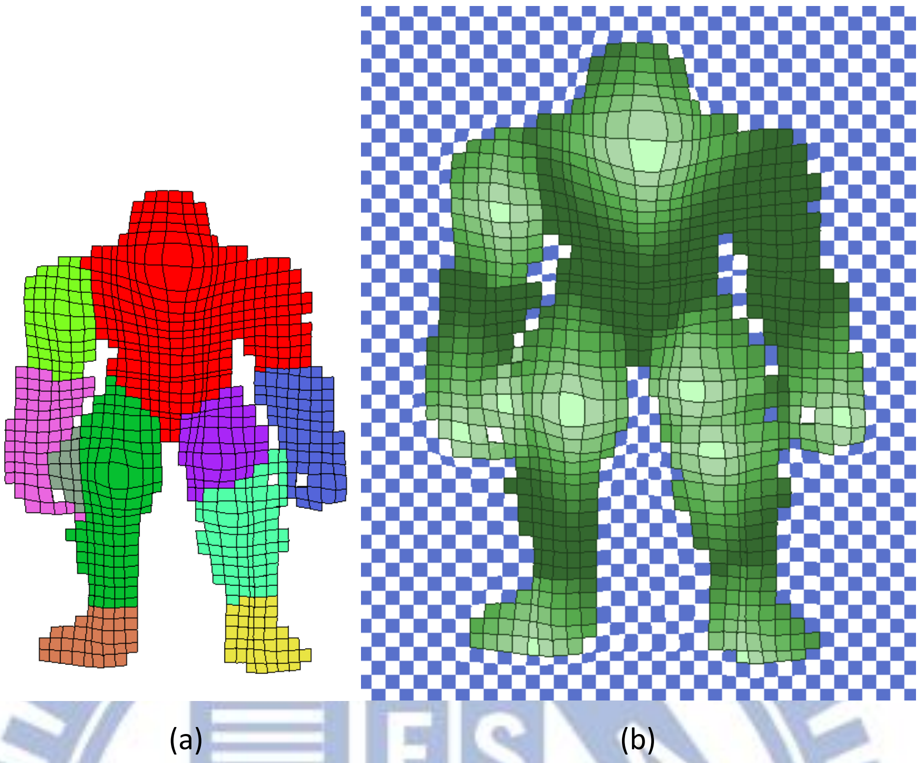


Figure 4.10: The rendered Op Art result of Hulk. (a) The clustering result. (b) The experiment result.

The corresponding execution times of each Op Art output image were shown in Table 4.2. We observe that the number of object squares and fake object squares dominates the execution time. The greater the number is, the more the execution time it spends. Besides, our system spends most of the time finding the new intersections of each horizontal line and vertical line. Because the points of a square may alter after square deformation and square alignment, as shown in Figure 4.11. The red points in Figure 4.11(a) transfer to the red points in Figure 4.11(b) after square deformation and square alignment. We find the new intersections by calculating the distance between the points of each row and column in the user specified S_{Max}. The two points with minimum distance will be the new intersection. When S_{Max} is larger, the displacement of the point is greater. So the range of each row and column that we should check is wider, as shown in Figure 4.12. After we find all the new intersections, we can obtain the points in a square.

Table 4.2: The execution times of each experiment results.

Model	Number of Squares	S_{Max}	Cluster Number	Object Squares + Fake Object Squares	Execution Time
Bunny	20X20	$\frac{W}{N_{Hori}} \times 1.0$	5	217	0.761
	40X40	$\frac{W}{N_{Hori}} \times 1.0$	5	755	1.064
	40X40	$\frac{W}{N_{Hori}} \times 2.0$	5	755	1.418
Pig	50X50	$\frac{W}{N_{Hori}} \times 1.0$	7	764	1.1
	60X60	$\frac{W}{N_{Hori}} \times 1.0$	10	1065	1.574
Zebra	50X50	$\frac{W}{N_{Hori}} \times 1.0$	8	592	0.912
Bear	60X60	$\frac{W}{N_{Hori}} \times 1.0$	7	878	1.533
Hulk	50X50	$\frac{W}{N_{Hori}} \times 1.0$	10	1169	1.302

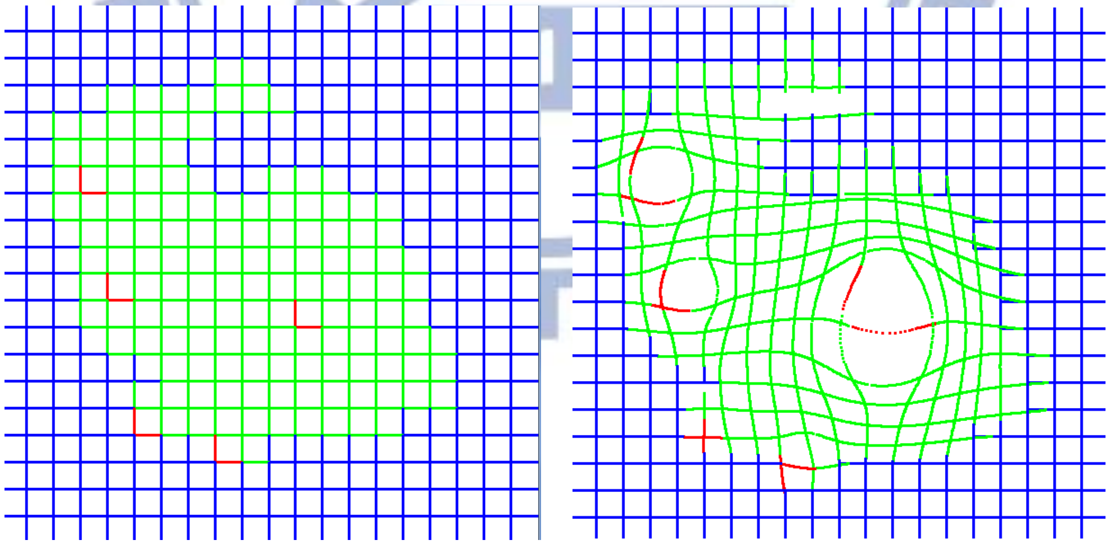

(a) (b)

Figure 4.11: The points of a square alter after square deformation and square alignment. (a) Before square deformation. (b)

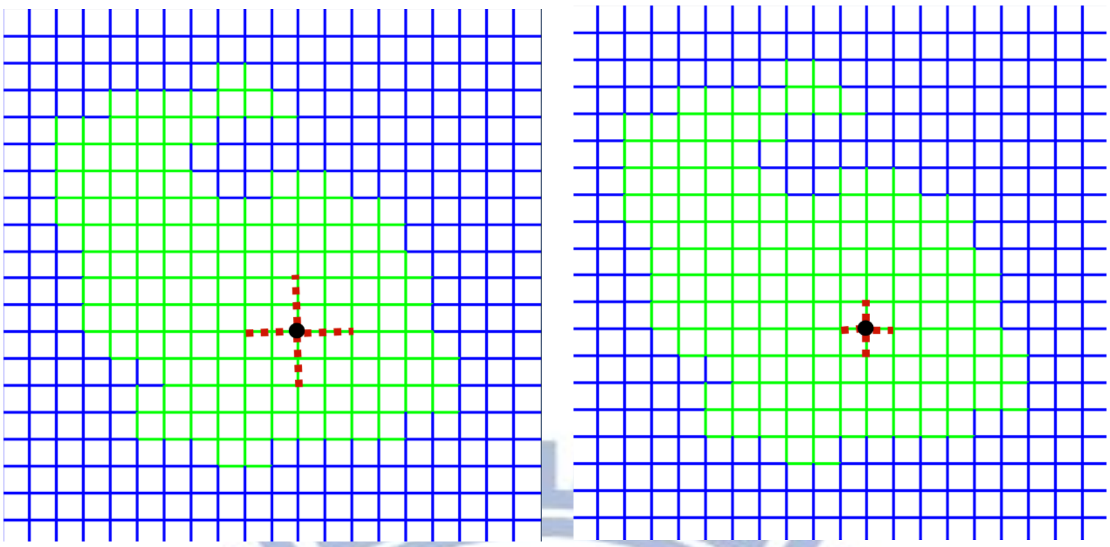


Figure 4.12: A new intersection is obtained by finding the smallest distance between the red points. (a) $S_{Max} = \frac{W}{N_{Hori}} \times 2.0$. (b) $S_{Max} = \frac{W}{N_{Hori}} \times 1.0$.

Chapter 5

Conclusion and Future Works

This thesis explores the square-based Op Art and the methods for achieving the desired artistic effects. The inflated and deformed squares create the swelling illusion which is similar to the artworks by Op artists. The input 3D models provide the depth information, so the users do not need to decide the inflated squares manually. We can cluster the squares by making use of the peak-climbing algorithm. Our system provides some parameters for the user preference. To imitate the Op artworks which have some deformed background squares, we create a type of square called fake object square. Besides, user can choose the rendered approach to tint the Op Art image. To avoid too many static squares existing in our result, we perturb the points of each square by the depth value of the adjacency squares. One limitation of our approach is that if the squares which have the smaller depth value are not the raised region of the input 3D model, it would fail to produce a reasonable result.

There are a few directions for further research. Our system differentiates the foreground and background by the rendered approach which tints the two regions by different colors. If we tint the two regions by the same color, the contour of the model may become unclear or disappear. Ideally, the system should identify areas that require greater deformation such as the finer details in an image. We expect it can help to create the illusory contour on the boundary of the model. Besides, we expect the technique introduced in this paper can be extended to create the animation of Op Art.

References

[1] Arad, N., et al. (1994). "Image warping by radial basis functions: Application to facial expressions." CVGIP: Graphical models and image processing **56**(2): 161-172.

[2] Bach, M. (2005). "Pyramid Illusion." http://michaelbach.de/ot/lum-pyramid/index.html.

[3] Chang, G. "Curvature-based Pen and Ink Drawings from 3-D Polygonal Objects."

[4] Chi, M.-T., et al. (2008). Self-animating images: illusory motion using repeated asymmetric patterns. ACM Transactions on Graphics (TOG), ACM.

[5] Dodgson, N. A. (2008). Regularity and randomness in Bridget Riley's early Op art. Proceedings of the Fourth Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging, Eurographics Association.

[6] Dunham, D. (2010). Hyperbolic Vasarely Patterns. Proceedings of Bridges 2010: Mathematics, Music, Art, Architecture, Culture, Tessellations Publishing.

[7] Gopnik, A. and A. Rosati (2001). "Duck or rabbit? Reversing ambiguous figures and understanding ambiguous representations." Developmental Science **4**(2): 175-183.

[8] Hermann, L. (1870). "Eine erscheinung simultanen contrastes." Pflügers Archiv European Journal of Physiology **3**(1): 13-15.

[9] Inglis, T. C., et al. (2012). "Op art rendering with lines and curves." Computers & Graphics **36**(6): 607-621.

[10] Inglis, T. C. and C. S. Kaplan (2011). Generating op art lines. Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging, ACM.

[11] Kitaoka, A. (2002). "Akiyoshi's illusion pages." http://www.ritsumei.ac.jp/~akitaoka/index-e.html.

[12] Kitaoka, A. (2006). "Anomalous motion illusion and stereopsis." Journal of Three Dimensional Images (Japan) **20**(4): 9-14.

[13] Moorhouse, P. (2003). "A dialogue with sensation: the art of Bridget Riley." Bridget Riley, Tate Publishing: 11-27.

[14] Parola, R. (1996). Optical art: theory and practice, Courier Dover Publications.

[15] Petry, S. E. and G. E. Meyer (1987). The perception of illusory contours, Springer-Verlag Publishing.

[16] Riley, B., et al. (2003). Bridget Riley, Tate Pub.

[17] Russell, S. (2009). "Artificial Intelligence: A Modern Approach Author: Stuart

Russell, Peter Norvig, Publisher: Prentice Hall Pa." 122.

[18] Seitz, W. C. (1965). The responsive eye, Museum of Modern Art.

[19] TroncosoΩ, X. G., et al. (2005). "Novel visual illusions related to Vasarely'snested squares' show that corner salience varies with corner angle." Perception **34**: 409-420.

[20] Tsofe, A., et al. (2010). "Chromatic Vasarely effect." Vision research **50**(22): 2284-2294.

[21] Ware, C. (2013). Information visualization: perception for design, Elsevier.

[22] Zanker, J. M. and R. Walker (2004). "A new look at Op art: towards a simple explanation of illusory motion." Naturwissenschaften **91**(4): 149-156.