Journal of Science and Engineering Technology, Vol. 2, No. 4, pp. 15-25 (2006)

Procedural Silhouette Edge Drawing

YU-CHEN HU and SHIH-KAI CHUNG
Graduate School of Multimedia and Animation Arts, National Taiwan University of Arts
59, Sec. 1, Daguan Rd., Banciao, Taipei County, Taiwan

ABSTRACT

In the development of computer graphics, non-photorealistic rendering (NPR) technology has been a focus of research in recent years. In comparison with photo-realistic rendering which simulates realistic natural images, NPR provides more concise and humanistic expressiveness in images. Among the many types of workflow in NPR technology, the procedural type provides the user an easier and simplified workflow. However, procedural NPR algorithms usually produce uniform and non-humanistic images due to a lack of interaction with the user.

Since silhouettes are an important feature of edges for expressing the shape of an object, our work takes silhouettes as examples to propose a stroke construction workflow based on edge topology wherein users can combine pure stroke elements according to user-defined rules for their formation. Thus, more humanistic expressions can be added to the resulting silhouettes. First, the procedural NPR framework for stroke generation is outlined. Then, the mainstream NPR techniques and their respective theorems are discussed, followed by an introduction to the framework and algorithm of the proposed system and a discussion thereof. Finally, the feasibility of the proposed framework is illustrated by a sample algorithm, after which pertinent research results are presented.

***Key Words*: **non-photorealistic rendering (NPR), procedure, feature edge, silhouette edge, pure stroke element, topology

程序性的輪廓線繪製方法

胡宇辰　鐘世凱
國立台灣藝術大學多媒體動畫藝術研究所
台北縣板橋市大觀路一段 59 號

摘　要

在電腦圖學的發展過程中，非寫實性算圖的技術在近幾年受到相當程度的重視。相較於寫實性算圖以模擬真實的自然影像爲目的，非寫實性算圖提供更爲簡練、且更具人性化的圖像表達方式。而在非寫實性算圖的操作流程中，程序性的操作方式雖然提供了使用者較爲簡便的操作流程。然而，由於程序性的演算流程中缺乏與使用者之間的互動，最終產生的影像仍不免會有過於制式且缺乏人性化等缺點。

由於輪廓線是描述物體形體的重要特徵線，因此本論文將以輪廓線爲例，提出一套以線條拓樸爲基礎的筆畫建構流程。透過使用者建立筆畫規則以組合基本筆畫單元，賦予輪廓線較具人性化的表現方式。本文首先簡述程序性非寫實性算圖架構，接著文獻探討部分將針對已發表之非寫實性算圖的基本演算原理以及非寫實性算圖中線條式樣逐一探討，然後詳述本論文所提出的程序性非寫實性算圖架構與其演算法，最後並以實做及相關成果佐證論文的研究。

關鍵詞：非寫實性算圖，程序性，特徵線，輪廓線，單純筆畫單元，拓撲

I. INTRODUCTION

Comparing with researches which work on simulating realistic nature image, studies in the field of NPR provide viewers better novel visual outlook on computer-generated images. In recent years, researchers have developed numerous theorems and algorithms to analyze and construct relative visual elements, such as presentation of feature edges, new shading methods, media simulation, and etc. Up to date, most NPR technologies emphasis on the precise or realistic simulation of visual elements, but leave out the important creation segment which can make arts more humanistic. Take researches in the field of line-drawing as examples, many researches were devoted to solve the questions about "Which lines should be drawn" rather than "How to draw these lines".

In comparison with most published NPR works, this paper will propose a more flexible framework, which can assist users to establish appropriate stroke elements. For silhouette drawing, the rule setting of line-drawing process is made as an independent stage in this framework, thus provides the user a flexible way to design different drawing styles. To better show the capability and flexibility of our procedural framework, a particle algorithm is embedded to create a variety of drawing style, as the results of research achievement.

1. Motivation and Purpose

Although procedural NPR provides users a convenient operation workflow, users can't intervene the stroke description during the executing process. This will result in an excessively stiff and humanistic-lacking rendering image. For the visual representation to show perfect humanistic effect, a procedural NPR framework can have various CG-related research fields included. However, the resulted framework, in general, has low executing efficiency due to the complexity. In consideration of such a problem, our work provides lighter and more flexible line-drawing constructing principles. Thus users can develop suitable line-drawing module according to this framework.

2. System Overview

Establishment of pure stroke elements and design of paint stroke elements both are important processes which can vitalize silhouette line-drawing. The system proposed by this paper works mainly on extracting stroke element through analyzing silhouette edges topology. By means of dissolving and recombining polygon edges topology information, this system establishes "pure stroke element" from the basic silhouette edges geometry information. Then, these "pure stroke element" are linked to each other, according to the calligraphic style rules, to form the "paint stroke element". Finally, this framework uses particle system, which can provide rich and colorful visual effect, to process stroke style simulation. Figure 1 shows the system workflow.

II. RELATED WORKS

1. Survey of NPR

In the history of computer graphics, rising and flourishing on photo-realistic rendering (PR) algorithm development has accomplished many astonishing visual effects. No matter they are raytracing, radiosity, or other global illumination methods, these algorithms have a common purpose to simulate all kind of nature phenomenon or objects realistically. Achievements can be seen in a variety of application fields, especially in digital entertainment field, such as games, TV commercials and movies. Since PR is able to provide realistic image information, a basic question must raise up: Why need to develop NPR technology? The answer mainly comes from two aspects – structure representation and visual representation.

A. Structure representation – PR can provide rich and realistic image information. However, not all visual element information of an image is required in all occasions. Abundant image information sometimes implies a negative meaning: Images can't represent specific focuses, particularly these focuses are what users really needed. Sometimes users persecute with rich image information, this often occurs when users thirst for a concise and clear structure information. For instance, structure expositions in manual of common home electric appliances are usually drawn in black lines with perspective. Clear illustration of an object can make users better understand the structure of object rather than using photorealistic image.

B. Visual representation – In the progress of art development, human compose art creations with all kinds of medias and achieve many great works. Today, computer can generate very realistic images, development of algorithm on how to simulate traditional paint skills has become an oncoming tide. In addition to simulating traditional media, some newly risen art fields also need NPR technology to simplify production workflow. This is particularly true in the field of traditional animation, and many researchers have proposed all kinds of novel algorithms for cartoon shading.

According to different geometry structures, feature edges can be summed up to three types [4]: silhouette, boundary and

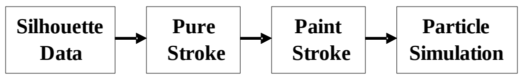

Fig. 1. System workflow chart

crease. Base upon model construction method (polygons or spline surfaces) the definition of each feature edge type will also be different. Since polygons are the basic element of modeling and rendering, in this paper our system framework will be implemented on polygon based models. Owing to the view-dependent feature edge information is related to camera parameters, this implies that since camera parameters have different settings (this may be distance between object and camera、camera's FOV value or else), silhouette information may changes even extracting from the same object. Silhouette data is different from fixed boundary and crease which are view-independent. Silhouette edge data will change, as camera or object moves. This property certainly drives many feature edge algorithms to emphasize on silhouette data processing [1, 9, 11, 13-15].

Many literatures have demonstrated that appropriate line stroke can represent object's inherent geometry structure [4]. In addition to geometry structure, other line properties such as color and width can also supply fairly helpful information. For example, visual effects can be conveyed by different line width in three situations [5]: Line-drawing with the same line width only conveys the construction on object itself; Enhancing outline width can extract object from background; Different width variation can construct perspective visual effect.

2. NPR Development Technology

Isenberg et al. [8] classify silhouette edge algorithms of NPR technology into three categories: Image-space based algorithms, object-space based algorithms and hybrid algorithms. Image-space based algorithms exploit image-buffer data as the main source information. These algorithms extract discontinuous data which usually represent the location where feature edges exist from the image-buffer. Object-space based algorithm make use of all the information obtained from 3D space as source information. Silhouette edges detection and extraction are both executed in 3D space. Different from image-space based algorithms and hybrid algorithms, object-space based algorithms will not over rely on information in image-buffer. Hybrid algorithms first use object space data to modify object geometry appropriately, then extract feature edges data from image-buffer. The purpose of geometry modification is to stand out relative feature edges data in image-buffer.

A. Literatures about image-space based NPR

Saito and Takahashi [16] address that profile (outline silhouette) and internal edges (inner silhouette) can be extracted by calculating first-order differential and second-order differential of depth map. Discontinuous portion of first-order differential calculation will reveal profile, and discontinuous portion of second-order differential will show up internal edge.

Deussen and Strothotte [3] provide another application on depth map. They use basic primitives to represent leaf (for instance, shape of disk), and calculate depth difference value between adjacent pixels. If depth difference value exceeds a certain threshold value, these pixels will be drawn. Just using depth difference value (equal to zero-order differential discontinuous calculation) will receive fairly good effect due to the leaf is individual object. No demand for first-order or second-order discontinuous calculation to find feature edges.

Hertzmann [7] suggests constructing normal map to complement boundary and crease information which can't be extracted from differential calculation of depth map. The RGB value of each pixel in normal map corresponds to the XYZ value of a normal vector. The main problem in depth map technology is that depth map can't calculate feature edges accurately if depth variation between polygons or objects are not manifest. Thus it will need a normal map to supply additional information to solve this problem.

B. Literatures about object-space based NPR

Although each type of feature edge has explicit geometry definition, for object-space based NPR technology to examine all edges one by one is very inefficient. Hence in the field of object-space based NPR researchers put their efforts on how to find out feature edges as fast as possible. Owing to silhouette edges have view-dependent property, silhouette edge detection naturally becomes the key point in this field.

Buchanan and Sousa [1] address edge buffer data structure to accelerate silhouette edge detection. In order to build up edge buffer data, initially each vertex must have a unique vertex index, and for each edge two additional flags are assigned: F flag (front-facing) and B flag (back-facing). After simple Boolean evaluation F flag and B flag will be set, and silhouette edge and boundary edge data can be obtained by evaluating each edge's F and B flag. The advantage of Buchanan and Sousa's edge buffer data structure is that it can fast sift out back-facing polygons which needn't to be conveyed to rendering pipeline. However, this method has a drawback that it can't find out crease data automatically. Extra artist flag must be assigned to edges manually to indicate which edge should always be drawn.

Gooch et al. [5] provide a software assisted algorithm to search for silhouette edges, by checking the projection region on the Gauss sphere which is formed by the neighbor polygon normal on each side of edge. If projection region intersects with virtual view plane (a plane parallel to view plane and passing through Gauss sphere center) then silhouette edge exists. Moreover, in order to speed up searching, Gooch et al. store projection region data in a hierarchy sphere. Using hierarchy sphere means representing sphere geometry in different hierarchy. Octahedron or icosahedron may be used as the initial geometry structure, and each next level sphere geometry will split current level geometry triangle face to form a finer sphere-like geometry.

Sander et al. [15] suggest an algorithm by combining hierarchy search tree and anchored cone method to accelerate silhouette edge searching. Construction of hierarchy search tree data structure is similar to construction of Huffman tree by manipulating “parent”, ”adopt” and “merge” operations between nodes. This is a bottom-up data construction method. Anchored cone then defines a non-closed cone volume for each node containing face cluster. Through setting a conservative bound face clusters, pure front-facing or back-facing can be determined. Sub-nodes testing is unnecessary if pure front-facing or back-facing face clusters are found, hence this method is helpful in economizing silhouette edge determination time.

C. Literatures about hybrid NPR

Hybrid NPR technology with respect to object-space based NPR technology has a critical difference: Hybrid NPR technology don’t have feature edge detection. In other words, edges on an object will not be classified to certain feature edge type (silhouette, boundary or crease). Hybrid NPR algorithms tend to operate on total [12] or easy chosen [13] polygons rather than execute time-consuming feature detection processing. The geometry structure will be modified to make feature edge data more obvious in image-buffer.

Raskar and Cohen [13] observe that silhouette edges only exist in first layer polygons (front-facing polygon) and second layer polygons (back-facing polygon), and address two methods to produce silhouette by modifying polygons of object: First method is shifting back-facing polygon toward camera with a certain distance. Depend on the dihedral angle between front-facing and back-facing polygon, silhouette edges with different width will appear. Second method is extending edges of back-facing polygons outward. Enlarge the back-facing polygons to be greater than front-facing polygons, then silhouettes will appear. By appropriate parametric control the second method can produce uniform width silhouette strokes.

On account of the advantage that OpenGL is hardware-supported, Gooch et al. [5] employ “PolygonOffset” function supplied by OpenGL (GL 1.1) to create silhouettes. PolygonOffset function shifts polygons according to the slope relative to near clip plane and far clip plane. Polygons with large slope imply that depth offset is great, and result in local discontinuous depth value. These portions are usually where feature edges occur.

Raskar [12] operates different types of feature edge creating process on all polygons without knowing what kind of feature edge a polygon can produce first. For each polygon Raskar judges whether it is front-facing polygon or back facing polygon, if back-facing polygon then extends polygon to make silhouette appear. For front-facing polygon two operations will be applied to reveal crease (Raskar classifies crease as ridge crease and valley crease; ridge crease is form by convex polygons and valley crease is form by concave polygons), and additional geometry structure (such as strip polygons) will let creases emerge.

3. Style Line-Drawing Technology

Line-drawing styles usually are not directly related to NPR technology type, even though different NPR technologies may limit line-drawing style (for instance, stroke style can be defined by user or not, stroke can be parameterized or not). By rebuilding stroke process line-drawing may violate the limitations caused by different NPR technologies being selected. However, stroke rebuilding process can provide additional advantage. This algorithm will not be embedded into each NPR framework, owing to stroke rebuilding technology has to consider style representation, and stroke sorting also consume extra executing time.

Both image-space based NPR and hybrid NPR technologies include stroke style representation inherently (this is because both technologies contain image-space processing). Without involving image-space, object-space based NPR technology simply extracts feature edge data, and stroke style representation will has a separate processing stage. Line-drawing style of image-space based NPR technology highly relies on the algorithm chosen to be operated on image-buffer data. Because each pixel is dealt with the same operation, the rendered image usually shows uniform line-drawing style. This monotonous style is good for structure representation but unfavorable to art representation. Hybrid NPR technologies can modify object geometry structure appropriately in object space, hence are able to produce more artistic line-drawing styles.

Stroke rebuilding technology produces more flexible visual representation. For NPR style animation algorithms, stroke rebuilding is an important stage to maintain frame-to-frame coherence [10]. In order to offer feature edges with stroke style, two mainstream technologies are applied currently: Skeleton Stroke Building System and Particle Stroke Building System. Northrup and Markosian [11] provide a workflow of skeleton stroke construction. Basically it constructs polygons along stroke edges, and after a series of image-processing stage stroke style effect can be generated. Particle Stroke Building System employs particle systems to spread particles along stroke path. Curtis [2] calculates “force-field image” according to the depth variation rate in depth map, and uses particle system to apply stroke style along the force-field.

According to difference in purposes and operation methods, different stroke style building strategies may be accepted. For example, the purpose of using NPR technology is for object structure representation or artistic creation? NPR program executing need is real-time or not? Final production is static image or animation? Different stroke style building systems have different limitation and suitability: Over abundant visual representation will obstruct structure information conveyed by object. Real-time NPR operation have to consider limitation on hardware, and animation production must prevent confusing glitter problem. All these factors have to be considered before designing a NPR algorithm.

III. INTEGRATE SILHOUETTE INFORMATION

The framework we proposed has two main stages in the workflow. The first stage is silhouette edge information integration (including silhouette edge extraction, structured silhouette edge, silhouette edge group construction and silhouette edge filter process), and the second stage is stroke style simulation. The purpose of the first stage is to create clear and systematic silhouette edge information, hence this stage will supply additional properties to silhouette edge in order to provide more flexible algorithm designing strategy.

1. Silhouette Edge Extraction

The first stage of our framework is silhouette edge extracting, and the purpose of this stage is to extract silhouette edge from model with the methods derive from basic definition of silhouette edge. Based on different NPR technologies a variety of algorithms can be chosen in this stage.

A. Description

Before drawing up a framework of a NPR algorithm designing case, designers must make sure the NPR technology about to use first (image-space based NPR technology, object-space based NPR technology or hybrid NPR technology). This is because designers will have to face the problem of silhouette edge extraction. A variety of algorithms can be chosen based on individual NPR technology. No matter which algorithm is chosen, basically in this stage silhouette edge must be extracted, and each silhouette must contain a unique edge index for identifying individual silhouette edge. This index may be inherent from model geometry data assigned by software, or properties supplied by designers (for instance silhouette edges integrated in image-buffer are assigned edge index by designers).

B. Algorithm analysis

In order to maintain the visual integrity of representation, either silhouette edges or boundary edges will be extracted following the convenience of algorithm design at the same time. Since calculation of silhouette edge and boundary edge can be put into practice with simple concept, a brute-force algorithm will be chosen to extract silhouette and boundary edges information in our system. For silhouette edge a strategy will be supplied to determine whether it is shared by front-facing polygon and back-facing polygon or not. Judgment of boundary edge can be applied if an edge is not shared by two polygons. Additionally, functions provided by MaxScript can supply both side polygon information connected to an edge (Figure 2).

2. Structured Silhouette Edge

After extracting silhouette edges from geometry model, related information has to be integrated since a well-defined data structure will help program framework to be more clear and methodical.

A. Description

Except edge index information, extra information should be obtained to realize connection condition of silhouette edges. According to the API functions supplied by a variety of software packages, additional information may be got by direct calling functions, or designers will have to program an algorithm to achieve this process. These extra information will then be integrated as a compound data structure. In addition to index of each edge itself, edge indices connected to

```
For each edge of an poly_object
{
   find Edge_Polygons
   If (number of Edge_Polygons equal to 1)
      Edge is Boundary Edge
   Else
   {
      N1 = Normalized Polygon1 Normal
      N2 = Normalized Polygon2 Normal
      V = Normalized View Direction
      If ((Dot N1 V) * (Dot N 2V) < 0)
         Edge is Silhoustte Edge
   }
}
```

Fig. 2. Silhouette and boundary edge extraction

current edge are recommended at least because recording of neighbor edges indices can enhance edge's connection sorting time.

B. Algorithm analysis

Algorithms afterward will emphasize on silhouette edge processing. Because boundary edges are simple closed loop in general cases, and silhouette edges usually form complex topology, relative analysis processes have to be executed until paint stroke path construction. To the setting of silhouette edge data structure, more additional properties will be included practically. According to the afterward strategy chosen by designers, different properties will be selected and their values will be set in afterward process.

Except Edge_Index property which stores current edge index, Side1_Connect_Edge_Index and Side2_Connect_Edge_Index save edge indices that connect to the current edge in each end-point. With Side1_Connect_Edge_Index and Side2_Connect_Edge_Index specified silhouette connected to each other can be found easily. Vertex1_Pos and Vertex2_Pos store end-point vertex position in camera space coordinate. Position information can be applied to curve construction directly but in our system position data in camera coordinate will be processed to get depth shifted position data. Vertex1_ZShift_Pos, Vertex2_ZShift_Pos save the position of vertex which is shifted to the same depth according to which group it belongs to. Shifted position information will be helpful in silhouette edge filtering process since silhouette edge with great length may provide trivial visual effect in the final presentation. Side1_Connect_Edge_Angle and Side2_Connect_Edge_Angle will store the angle value with each side of neighbor silhouette edge after shifted, because silhouette edge link with sharp angle will be cut to form a more pure geometry element.

3. Silhouette Edge Group Construction

With structured silhouette edge data constructed, silhouette edges that connect to each other can be classified to the same group. Later in silhouette edge filtering stage a group will be taken as a basic processing element. Sorting within the same group will be much faster than sorting overall silhouette edges.

A. Description

According to the structured silhouette edge information, silhouette edges connected to each other can be assigned to the same group, thus each silhouette edge group is a non-broken silhouette edge collection. The purpose of grouping is to isolate disconnected silhouette information, and provides a preliminary classifying on silhouette edge topology. Furthermore, grouping will also make some silhouette edge searching process more efficient (take group as a base element to execute sorting and searching process).

B. Algorithm analysis

Silhouette edge grouping employs vertex index of the current edge as the sorting index, and searches others with the same vertex index on their side, thus silhouette edges connected to each other will be assigned to the same group (Figure 3).

4. Silhouette Edge Filtering

In order to simplify silhouette edge data, each silhouette edge group will execute a filtering process. Filtering processing not only reduces the silhouette edge data amount, but also eliminates insignificant silhouette edges and makes final constructing curve smoother.

A. Description

The filtering process treats each silhouette edge group as a base element to choose appropriate silhouette edges. For models with fine details this stage provides an obvious advantage in reducing data amount. In addition, for object-space based NPR technology, well designed algorithms can moderate the appearance of sharp-tooth silhouette edge, and this is helpful to construct clear silhouette strokes. Silhouette edge filtering will modify silhouette edge geometry information, thus this stage could be seen as an edge reconstruction process.

B. Algorithm analysis

A more intuitive process of silhouette edge filtering will be chosen in our system. Silhouette edges in the same group will be shifted to the same depth with no position variation on image plane (Figure 4), and shifted position will be stored individually for later use. Actual model geometry will not be modified, and shifted positions are obtained just by position

```
For each Struct_Silhouette_Edge(SSE)
{
   Current_Vertex1_Index = SSE.Vertex1_Index
   Current_Vertex2_Index = SSE.Vertex2_Index
   For each SSE except itself
   {
      If ((SSE.Vertex1_Index = = Current_Vertex1_Index) or
          (SSE.Vertex2_Index = = Current_Vertex2_Index))
      {
         Group Silhouette Edge
         Del/mark grouped Silhouette Edge in SSE_Collection
      }
   }
}
```

Fig. 3. Pseudo-code of silhouette edge grouping

```
For each Silhouette_Edge_Group
{
    Find Max_Z_Depth
    For each SSE in Silhouette_Edge_Group
    {
        Current_X = SSE.Vertex_Pos.X
        Current_Y = SSE.Vertex_Pos.Y
        Current_Z = SSE.Vertex_Pos.Z
        Z_Offset = Max_Z_Depth – CurrentZ
        Cur_Vector = Normalized[Cur_X, Cur_Y, Cur_Z]
        XZ_Vector = Normalized[Cur_X, 0, Cur_Z]
        YZ_Vector = Normalized[0, Cur_Y, Cur_Z]
        X_angle = Acos (Dot YZ_Vector Curt_Vector)
        Y_angle = Acos (Dot XZ_Vector Cur_Vector)
        X_Offset = Z_Offset * Tan (X_Angle)
        Y_Offset = Z_Offset * Tan (Y_Angle)
    }
}
```

Fig. 4. Vertices on silhouette edge shift to the same depth

value calculation.

For silhouette edge length information, 3D space represents totally different meanings from 2D image space. Owing to the final result is a 2D image, a viewpoint from image space has to be concerned to make filtering process more accurate. Practical coding in our system is filtering silhouette based on a length threshold value set by designers, discarding silhouette edges which are under threshold value, and re-connecting neighbor silhouette edges (Figure 5).

IV. STROKE STYLE REPRESENTATION

Stroke style simulation is the second stage of the framework. The processes include pure stroke data structure, analyze pure stroke, paint stroke data structure, and particle system simulation. Integrated silhouette edge information can be categorized to four kinds of pure stroke types. According to the rules of calligraphic style, pure stroke will be organized in turn in order to construct paint stroke. Finally particle system will be applied to simulate media effect. Particle system supplies a sufficient flexible properties to represent visual effect and physical phenomenon of media. This is the reason that particle system simulation is an independent processing stage in the framework.

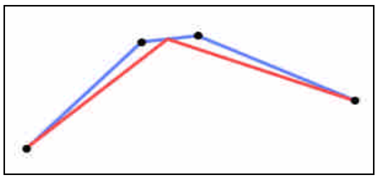

Fig. 5. Silhouette edge filtering

1. Pure Stroke Data Structure

For stroke style representation our method analyzes each silhouette edge group to obtain basic stroke elements according to specific silhouette edge connecting condition. Later these basic stroke elements will be organized based on stroke painting order to form a more integrated line-drawing construction.

A. Description

Pure strokes can be concluded in terms of silhouette edge topology, and deemed as base stroke elements. Each pure stroke includes three properties:

1. Pure stroke is a single path stroke.
2. No self-intersection will occur to pure stroke except looped pure stroke.
3. Sharp angle will not be found in pure stroke.

According to pure stroke's properties, four types of connection situation of pure stroke's two-side vertex can be categorized：

1. S Type: Vertex doesn't connect to other pure strokes (Figure 6(a)).
2. A Type: Vertex connected to another pure stroke with a sharp angle (Figure 6(b)).
3. C Type: Vertex connected to multiple pure strokes (Figure 6(c)).
4. L Type: A pure stroke connected to its self (Figure 6(d)).

A pure stroke analysis example is presented in Figure 7.

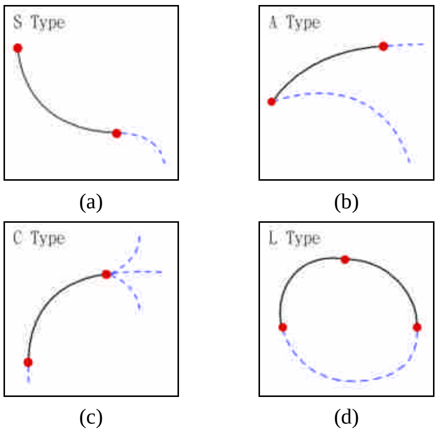

Fig. 6. Four types of connection situation of pure stroke

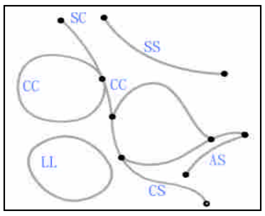

Fig. 7. A pure stroke analysis example

Data structure of pure stroke at least includes: current pure stroke's type, silhouette edge information contained in current pure stroke, and pure stroke information connected to each side of current pure stroke. Calligraphic style representation will be different for individual pure stroke type, such as a SS type stroke will be thin in the beginning and ending portion, thus current pure stroke's type have to be recorded. Silhouette edge information contains vertex position and angles of each connected silhouette, and this information will be used in later stage for curve construction. For fur referencing total silhouette edge information contained in current pure stroke should be included. Finally, adjacent connected pure stroke index will be recorded for faster sorting.

B. Algorithm analysis

Before constructing pure stroke data structure, each silhouette edge group has to be divided as pure stroke. Based on pure stroke connection situation, compound types of A, S, C (AA, SS, CC, AS, SA, AC, CA, SC, CS) and LL type can be classified (Figure 8). When pure stroke type and silhouette edges included by a pure stroke are confirmed, relative silhouette edge data will be integrated to the newly constructed pure edge data structure.

2. Analyze Pure Stroke

After obtaining basic stroke elements, the factor of stroke's trend has to be concerned to connect all basic stroke element links. In analyzing pure stroke stage algorithms will be applied to join pure strokes in certain order, just as simulating painters' drawing strokes in order.

A. Description

This stage sets rules of calligraphy style. Pure strokes will be connected to each other according to rules provided by designers to form painting stroke. Owing to strokes have direction, linking of pure strokes must take directional property into account.

```
For each Pure_Stroke in Silhouette_Edge_Group
{
   For each connect side of Pure_stroke do
   {   //SCPS=Side_Connect_PureStroke
      If (Pure_Stroke.SCPS_Index_Count = = 0)
         Add PureStroke_ PureStroke_Type "S"
      Else if (Pure_Stroke. SCPS_Index_Count > 1)
            Add PureStroke_ PureStroke_Type "C"
      Else if (Pure_Stroke. SCPS_Index_Count = = 1)
         {
            If (Pure_Stroke.Vertex1_ZShift_Pos = =
               Pure_Stroke.Vertex_ZShift_Pos)
               Add PureStroke_ PureStroke_Type "L"
            If (Pure_Stroke.SCPS_Angle >
               Threshhold_Angle)
               Add PureStroke_ PureStroke_Type "A"
         }
   }
}
```

Fig. 8. Pure stroke classify process

B. Algorithm analysis

Construction of calligraphy style rules relies on researching of many kinds of media properties and art styles. A simpler algorithm is supplied to connect pure strokes hierarchically (Figure 9). To construct a hierarchical connecting stroke a main stroke has to be found first. Pure strokes intersect with main stroke are defined as level 1 stroke. Strokes connect with level 1 will be sorted out to define level 2 stroke and so on (Figure 10). Each pure stroke intersects with current level pure stroke will be defined as next level pure stroke, and this will form a stroke structure distributed as a tree structure.

3. Paint Stroke Data Structure

Since pure strokes' link orders have been decided, the next step is to organize pure stroke links into a more integrated data type - paint stroke. Paint stroke data structure contains the information for curve construction. It also can contain the parameters about stroke style representation if no additional stage is responsible for it.

A. Description

Final data structure can include stroke style information, thus paint stroke data structure is addressed to contain geometry

```
For each Silhouette_Group
{
   Collect all Pure_Stroke
   Find the longest pure stroke
   Remove chosen pure stroke
   While (PureStroke_Count != 0)
   {
      Trace Next_Level_Path if "C" type
      Remove chosen pure stroke
   }
}
```

Fig. 9. Hierarchical stroke construction

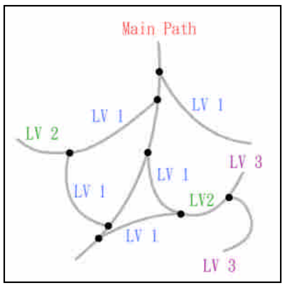

Fig. 10. Hierarchical stroke structure

information of stroke path and relative stroke style parameters. Paint stroke data structure is provided to supply a final integrated data structure. Stroke style effect established based on skeletal stroke building system is recommended to accomplish skeletal stroke polygon construction, and effect produced by particle system is suggested to complete stroke path curve construction.

B. Algorithm analysis

Algorithm provided in this stage emphasizes on obtaining usable information for stroke path curve construction. Spline construction in MaxScript only needs vertex position information, thus in practical algorithm designing vertex position of hierarchical pure stroke will be extracted in order, and integrated in a simple data structure. Paint stroke data structure which contains stroke style controlling properties has been described, but owing to particle system supplies more extensively control ability over stroke style, practical algorithm designed based on particle stroke building system can integrate stroke style controlled with particle system.

4. Particle System Simulation

To get various visual effects, particle system will be applied in the final stage in our framework. Since particle system possesses complex and flexible tuning ability, this will be helpful in assisting artist's creativity.

A. Description

Advantage of particle stroke building system is that particle system can provide simulation of physical phenomenon, and media reality can be improved by setting corresponding particle system parameters. Representation of media appearance can be achieved by assigning pre-produced texture map or procedural texture to particles.

B. Algorithm analysis

Many software packages have provided practical particle system tools, our algorithm only needs several basic parameters to test stroke style appearance: particle size, grow and fade speed of particle, and random size variation etc. In practical operation a Particle Flow tool supplied in 3DS Max 6 is used for stroke style testing (Figure 11 (a), (b), (c), (d)).

V. RESEARCH RESULTS

Strokes can represent a variety of styles with textures no matter texture maps are pre-produced by users or created by procedural method provided by available software packages. If textures are made by a procedural processing, then the procedural texture creating process can be integrated into procedural stroke construction scheme. For testing purpose a procedural texture called "Dent" supplied by 3DS Max is

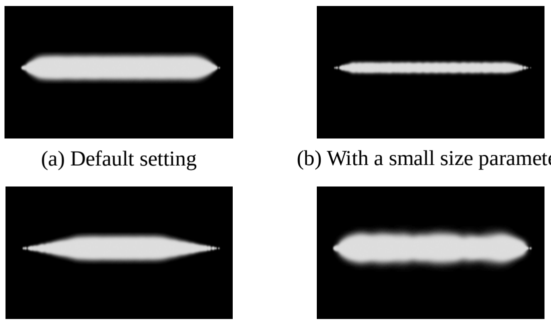
(a) Default setting
(b) With a small size parameter
(c) With a fade/grow parameter
(d) With a random size parameter

Fig. 11. Variation of stroke style with various parameter setting

chosen (Dent texture generates the effect just like noise does). In this section relative research results will be displayed to show that stroke styles and calligraphic styles can still be maintained in the procedural line-drawing process.

First exhibition is the teapot model. Figure 12(a) is the original model. In practical design a simple user interface contains two relative parameters about particle system: particle size and particle birth rate. Few parameter controls are provided owing to the complex particle effects will disturb observation of stroke style and calligraphic style variations.

Figure 12(b) represents the default setting of relative parameters, and Figure 12(c), (d), (e) display the effect of operating modification on particle size and particle birth rate parameter individually. Altering particle size parameter can influence stroke width, and modifying particle birth rate has apparent change in stroke density.

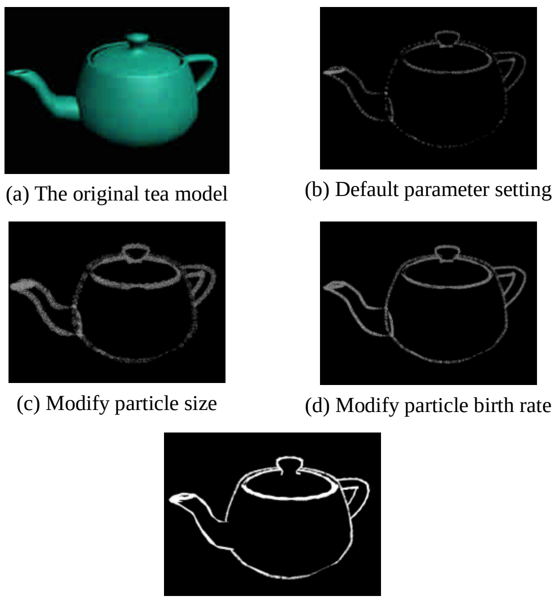
(a) The original tea model
(b) Default parameter setting
(c) Modify particle size
(d) Modify particle birth rate
(e) Increase birth rate

Fig. 12. Experiment results with teapot model

In the process of testing, an open boundary model is chosen to test whether the extracting process of boundary edges in practical algorithm works correctly. A mask model with many boundaries is selected (Figure 13(a)). With particle effects, pleased stroke style representation is appeared especially around the jaw of mask (Figure 13(b), (c)). Abundant visual effect can be presented with all kinds of texture map applied to particles. The final visual representation of line-drawing style greatly relies on tuning of particle system parameters and selection of texture map applied to the particles (Figure 13(d), (e), (f)).

Presentation of a series of images (Figure 14) describes the order of stroke painting. Each stroke's painting timing can be integrated into "Paint stroke data structure" to form a line drawing animation with specific order that is influenced by the algorithm designed in the "Analyze Pure Stroke" stage of our framework.

VI. CONCLUSIONS AND FUTURE WORK

1. Conclusions

For line drawing process in procedural NPR algorithms, besides emphasizing on line information accuracy and realistic imitation of media effect, in recent years researchers in the field of NPR prefer to think from the artist's point of view, and try to get new inspiration from the process of art creation behavior. This tendency also implies that accurate edge information and reality of media simulation still can't redeem the lack of humanistic representation of computer-generated images. This paper proposes an open structure to attach importance to this problem: For procedural NPR technologies with humanistic representation an algorithm must be constructed independently in the framework corresponding to the art creation behavior. Therefore the framework addressed by this paper makes the process of converting pure stroke to painter's stroke a stand alone stage. In the meanwhile, this framework also supplies definitions of pure stroke. This provides designers a basic element when establishing stroke style or calligraphic style algorithm.

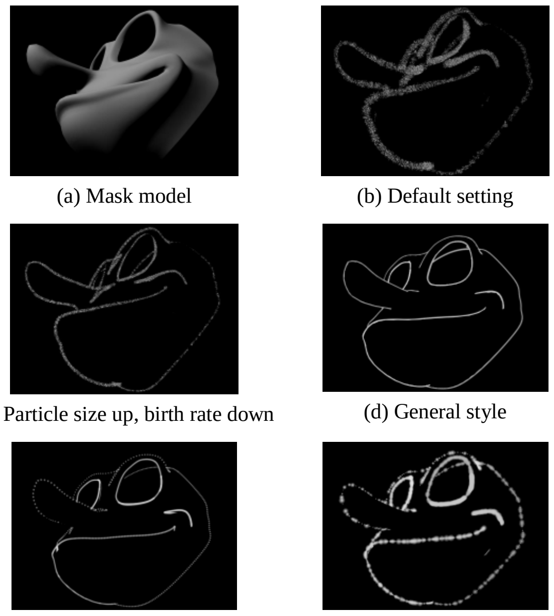

Fig. 13. Experiment results with an open boundary (mask) model

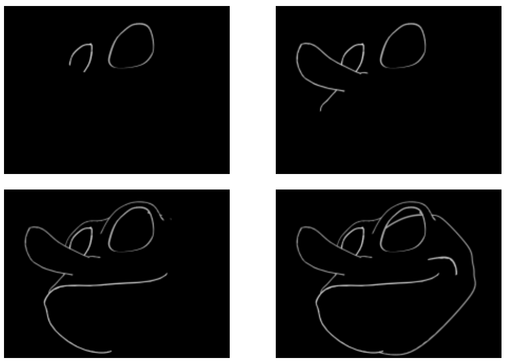

Fig. 14. Process of ordered line-drawing

Invention of pure stroke not only contributes to still NPR line drawing technology, but also is helpful in NPR animation algorithm development. In many situations silhouette edges connected to each other in 3D space are belong to the same object. Categorizing these silhouette edges to pure stroke as a basic element can improve time-consuming silhouette edge sorting process. Determination of coherence of line visibility can be speeded up if pure strokes are taken as a basic determination element [16]. Anyhow, pure strokes can provide more different thought to line drawing technology design.

2. Future Work

Examining our framework, there are four main stages to make NPR line drawing algorithm with different effects: algorithms of silhouette edge extraction, silhouette edge filtering, calligraphy construction and media simulation. Among these stages many researches have devoted to accurately represent information in silhouette edge extracting process and reality of media simulation. But methods to reduce silhouette edge information and ways to establish calligraphy style so far don't have much persuasive development. Practical algorithms in our system only design with simple thought. Silhouette edge filtering and calligraphy style establishing still need some further study. Especially calligraphy styles are different to each other according to a variety of media, tools and personal style. It is almost impossible to create a general purpose algorithms, particularly each person has individual drawing habit, not to mention that limitations in specific media or drawing style will strangle freedom of artist's creation. How to provide artists a convenient and simple user interface to establish calligraphy style or stroke style rules, and convey these rules to an general purpose framework to produce a unique line drawing procedure, is a new direction for NPR's future development.

REFERENCES

1. Buchanan, J. W. and M. C. Sousa (2000) The edge buffer: A data structure for easy silhouette rendering. Proceedings of 1st International Symposium on Non-Photorealistic Animation and Rendering, Annecy, France.
2. Curtis, C. (1998) Loose and sketchy animation. SIGGRAPH Conference Abstracts and Applications, Orlando, FL.
3. Deussen, O. and T. Strothotte (2000) Computer-generated pen-and-ink illustration of trees. *Computer Graphics*, 34(4), 13-18.
4. Gooch, B. and A. Gooch (2001) *Non-Photorealistic Rendering*, AK Peters, Natick, MA.
5. Gooch, B., P. P. J. Sloan, A. Gooch, P. Shirley and R. Riesenfeld (1999) Interactive technical illustration. Proceedings of Symposium on Interactive 3D Graphics, Atlanta, GA.
6. Grabli, S., E. Turquin, F. Durand and F. Sillion (2004) Programmable style for NPR line drawing. Proceedings of EUROGRAPHICS Symposium on Rendering Techniques, Norrköping, Sweden.
7. Hertzmann, A. (1999) Introduction to 3D non-photorealistic rendering: Silhouettes and outlines. *SIGGRAPH Course Notes*, Los Angeles, CA.
8. Isenberg, T., B. Freudenberg, N. Halper, S. Schlechtweg and T. Strotthotte (2003) A developer's guide to silhouette algorithms for polygonal models. *Proceedings of IEEE Computer Graphics and Applications*, 23(4), 28-37.
9. Markosian, L, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein and J. Hughes (1997) Real-time nonphotorealistic rendering. *Computer Graphics*, 31(4), 415-420.
10. Masuch, M., L. Schuhmann and S. Schlechtweg (1998) Frame-to-frame coherent line drawing for illustrated purposes. Proceedings of Simulation and Visualization, SCS Europe, Magdeburg, Germany.
11. Northrup, J. D. and L. Markosian (2000) Artistic silhouettes: A hybrid approach. Proceedings of First International Symposium on Non-Photorealistic Animation and Rendering, Annecy, France.
12. Raskar, R. (2001) Hardware support for non-photorealistic rendering. Proceedings of SIGGRAPH / EUROGRAPHICS Symposium on Graphics Hardware, Los Angeles, CA.
13. Raskar, R. and M. Cohen (1999) Image precision silhouette edges. Proceedings of Symposium on Interactive 3D Graphics, Atlanta, GA.
14. Rossignac, J. and M. Emmerik (1992) Hidden contours on a framebuffer. Proceedings of the 7th Workshop on Computer Graphics Hardware, Eurographics, Cambridge, England.
15. Sander, P. V., X. Gu, S. J. Gortler, H. Hoppe and J. Snyder (2000) Silhouette clipping. *Computer Graphics*, 34(4), 327-334.
16. Saito, T. and T. Takahashi (1990) Comprehensible rendering of 3-D shapes. *Computer Graphics*, 24(4), 197-206.

收件：**94.12.16**　修正：**95.02.24**　接受：**95.04.14**