Modeling and Rendering of Realistic Feathers

Yanyun Chen Yingqing Xu Baining Guo Heung-Yeung Shum

Microsoft Research Asia*

Abstract

We present techniques for realistic modeling and rendering of feathers and birds. Our approach is motivated by the observation that a feather is a branching structure that can be described by an L-system. The parametric L-system we derived allows the user to easily create feathers of different types and shapes by changing a few parameters. The randomness in feather geometry is also incorporated into this L-system. To render a feather realistically, we have derived an efficient form of the bidirectional texture function (BTF), which describes the small but visible geometry details on the feather blade. A rendering algorithm combining the L-system and the BTF displays feathers photorealistically while capitalizing on graphics hardware for efficiency. Based on this framework of feather modeling and rendering, we developed a system that can automatically generate appropriate feathers to cover different parts of a bird's body from a few "key feathers" supplied by the user, and produce realistic renderings of the bird.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; F.4.2 [Mathematical Logic and Formal Languages]: Grammars and Other Rewriting Systems; I.3.3 [Computer Graphics]: Picture/Image Generation. J.3 [Life and Medical Sciences]: Biology;

Keywords: L-system, bidirectional texture function, natural phenomena, rendering, feather, bird

1 Introduction

Since the earliest times, the natural beauty of feathers has fascinated humans – from Hawaiian chiefs adorned in brightly colored robes of feathers to Native Americans wearing exquisitely crafted headdresses. Feathers are everyday wear for birds, who use feathers to fly, to keep warm, and to attract mates. Despite their ubiquitous nature, we do not have a systematic way to model and render feathers in computer graphics. This work is an attempt to fill that gap.

There are two main tasks in feather modeling and rendering. The first is the modeling and rendering of individual feathers. At the macroscopic level, feathers come in different types and shapes: bristles, contour feathers, down, flight feathers, semiplumes, and filoplumes [11]. At the microscopic level, feathers have a special appearance which is attributable to their barbs and barbules [11] shown in Fig. 3. These fine-level visible geometry details constitute the mesostructure of the feather blade [6, 3]. The main challenges in modeling and rendering feathers are to allow the user to easily generate many feathers of different shapes and to capture the special appearance of feathers.

Figure 1: An eagle modeled and rendered using our system.

The second task is growing feathers on birds. Looking at a bird you can see that various types of feathers are arranged over the bird's body in an extremely ordered fashion. Feathers overlap each other, usually covering most of a bird's skin and thus creating a streamlined shape that is perfect for flight. Achieving this feather arrangement by manually placing thousands of feathers onto a bird's body is clearly very tedious and time-consuming.

In this paper, we present two techniques, one for each task described above. For modeling and rendering of individual feathers, our technique uses a bidirectional texture function (BTF) [3] controlled by a parametric L-system [14]. Consistent with feather anatomy [18], this L-system allows the user to generate feathers of different shapes by adjusting a few parameters. Another important feature of our L-system is that it simulates the random gaps in the vanes of feathers, which are important for visual realism.

The special feather appearance arises from both spatially-variant surface reflectance and local surface height variations due to the barb-barbule mesostructure. The mesostructure, for example, creates fine-scale shadows, occlusions, and specularities that are integral parts of the feather appearance [6, 3]. To capture this level of complexity, we developed a feather rendering algorithm based on the L-system and a realistic BTF [3] that models the barb-barbule mesostructure and the directional radiance distribution at each surface point. This BTF is pre-computed so that graphics hardware can be used to render feathers efficiently.

Our technique for feather growing allows the user to provide a 3D model of a bird and a number of "key feathers" at different

* 3F Beijing Sigma Center, No 49 Zhichun Road, Haidian District, Beijing 100080, P R China, email: bainguo@microsoft.com

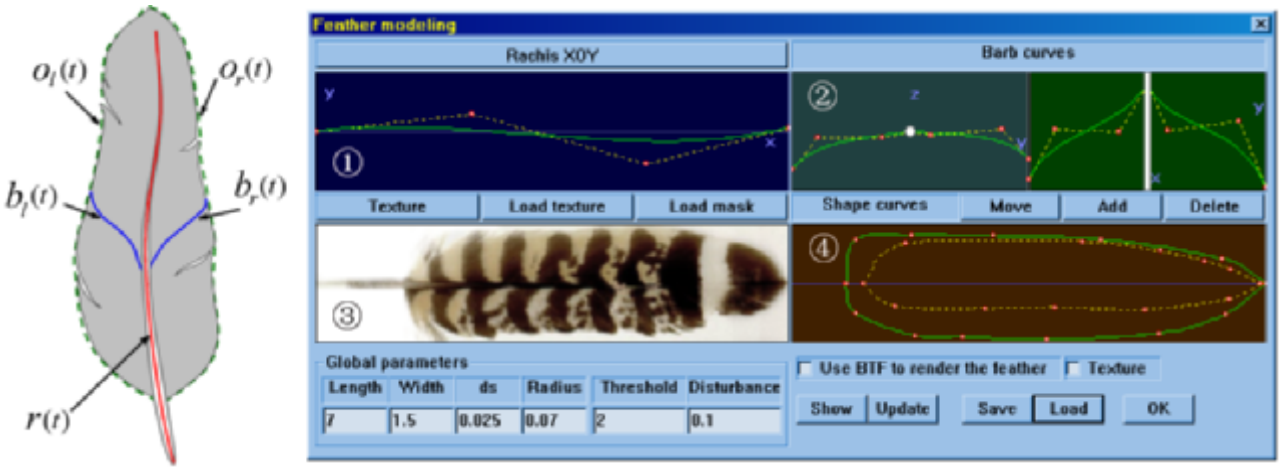

Figure 2: The user interface of our feather modeling system. The sketch on the left illustrates the rachis curve, the left and right barb curves, and the left and right outline curves. The user can interactively change these curves to control the overall shape of the feather. Window (1) is for controlling the rachis curve, window (2) for the left and right barb curves, and window (4) for the left and right outline curves. Window (3) is a user-supplied texture.

locations on the model. The feather growing algorithm then automatically generates all feathers on the model, guided by the shape and orientation hints from the user. The feather growing process includes creating a feather for the current location based on nearby "key feathers" and placing all feathers so created in an orderly fashion. The main challenge in growing feathers is determining the orientations of feathers such that neighboring feathers do not interpenetrate. We address this problem with a recursive collision detection technique.

Fig. 1 provides an example of bird rendering generated by our system. Applications of realistic feather modeling and rendering include display of birds and feather objects (Native Americans headdresses, feather ornaments, etc.) in motion pictures and games, especially those including birds or feather objects as main characters or objects of importance. Realistic rendering of birds and feather objects are also desirable for web-based education (e.g., virtual bird walk) or product display.

The rest of the paper is organized as follows. In Section 2, we review some related work. Section 3 discusses the modeling and rendering of individual feathers. Section 4 describes our system for placing feathers on a bird's body. In section 5, we show examples of feather modeling and rendering. We conclude with some discussion about future work in Section 6.

2 Related Work

A number of researchers have mentioned feathers in their papers. Dai et al. developed a method for synthesizing feather textures of a special class of feathers, i.e., Galliformes feathers [1]. Using chaotic structures, they modeled the "eye-like spot" and "river-like strip" for feathers in the Galliformes family. They also crafted an ad-hoc branching structure to model feather skeleton. Schramm et al. applied subsurface modeling technique to the study of the surface reflectance properties of an iridescent hummingbird feather [19]. Franco and Walter presented a parametric model for feather modeling based on Bezier curves [4]. Two nice examples are shown in [4], but no algorithm details are given.

L-systems have been used by Prusinkiewicz et al. extensively to capture the natural beauty of plants [13, 14, 15]. A plant is modeled as a linear or branching structure composed of repeated units called modules. An L-system represents the development of this structure by productions. In the original L-system [7], the modeled structure is a finite collection of modules and each module is in one of the finite number of states. Parametric L-systems increase the expressiveness of L-systems by adding a state vector of numerical parameters [5, 15]. The meanings of the parameters depend on the semantics of the module definition. For example, the parameters may describe the shape of a plant being modeled. Parametric L-systems are important for feather modeling because it allows the user to easily create a large number of feathers by varying the parameters. L-systems have also been used by Parish and Müller for modeling urban environments [10]. We are not aware of L-systems used for feathers or birds.

The BTF was introduced by Dana et al. for describing real-world textures [3]. A 2D texture is a poor description for real-world textures because it completely ignores surface mesostructures. The BTF, on the other hand, captures surface mesostructure well. Summaries of recent work on the BTF can be found in [2, 8, 20]. A concept related to the BTF is the polynomial texture map proposed by Malzbender et al. [9]. The polynomial texture map can model the appearance changes of real-world textures due to change of illumination, but the viewing direction must be fixed.

3 Individual Feathers

3.1 Feather Geometry

Fig. 2 illustrates the user interface for modeling feather geometry. We know from Fig. 3 that the geometry of a feather is defined by its rachis and the barbs distributed along both sides of the rachis. In our system, the shapes of the rachis and the barbs are mainly controlled by the rachis curve $\mathbf{r}(t)$, left barb curve $\mathbf{b}_l(t)$, right barb curve $\mathbf{b}_r(t)$, left outline curve $\mathbf{o}_l(t)$, and right outline curve $\mathbf{o}_r(t)$ as shown in Fig. 2. With these curves, the user has fine control over feather geometry, which is necessary for modeling various types of feathers. For example, a flight feather has a narrow leading edge that the wind hits first and a wider trailing edge. Flightless birds, on the other hand, have almost symmetrical sides on corresponding feathers. This difference in feather geometry can be captured by adjusting the outline curves $\mathbf{o}_l(t)$ and $\mathbf{o}_r(t)$.

For simplicity, we assume that the shapes of all barb curves on the same side of the rachis are identical except for their lengths.

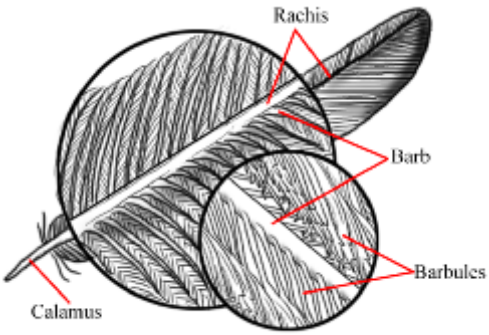

Figure 3: Feather anatomy. The rachis or shaft of the flight feather supports the vanes (i.e., the blades) of the feather. Within the vane of the feather, there are two lateral sets of barbs, interlocking the feather together. On both sides of a barb are many barbules.

Under this assumption the barb curves at the position $\mathbf{r}(t_i)$ along the rachis are $\mathbf{b}_l(u)$ and $\mathbf{b}_r(v)$ with lengths determined by $\mathbf{o}_l(t_i)$ and $\mathbf{o}_r(t_i)$ respectively.

We can regard a feather as a branching structure composed of repeated units called modules. An L-system represents the development of a branching structure by productions [14]. A production replaces a predecessor module by several successor modules. A production can either be context-free and depends only on the module replaced, or be context-sensitive, in which case the production depends on the replaced modules as well as its immediate neighbor modules. We use context-free productions of the form

$$id : pred : cond \rightarrow succ$$

where id is the production label, whereas $pred$, $cond$, and $succ$ are the predecessor, condition, and successor respectively. The production is carried out only if the condition is met.

Given the rachis, barb, and outline curves, we can model feather geometry using a parametric L-system [14, 13] as follows:

$$\begin{array}{l} \omega : R(0) \\ p_1 : R(i) : i < N \rightarrow [B_L(i,0)][B_R(i,0)]R(i+1) \\ p_2 : B_L(i,j) : j < M_L \rightarrow B_L(i,j+1) \\ p_3 : B_R(i,j) : j < M_R \rightarrow B_R(i,j+1) \end{array} \quad (1)$$

where N defines the length of the feather as well as the density of the barbs at each side of the rachis while M_L and M_R define the lengths of the left and right barbs respectively. We follow the notation of [14, 13]. The axiom $R(0)$ generates a feather based on the rachis and barb curves. Production p_1 produces a small segment of the rachis according to the rachis curve $\mathbf{r}(t)$ and grows a barb on each side of the rachis using recursion. Production p_2 creates a small segment of the left barb according to the left barb curve $\mathbf{b}_l(u)$ while production p_3 proceeds similarly on the right barb. The leftmost image in Fig. 4 shows a feather created using equation (1).

A problem with equation (1) is that it ignores the interaction between neighboring barbs. A feather geometry generated by equation (1) looks plausible but too regular. For a real feather, the two lateral sets of barbs within the vane of the feather interlock the feather together. The interlocking is important for flight, keeping the air from rushing right through the feather. When the interlocking system of a feather is disturbed, as when a twig brushes through a feather, random gaps form between the barbs on the same side of the rachis. This is a phenomenon that we want to capture qualitatively. On the one hand, neighboring barbs cling to each other by the little hooks called cilium on the ends of the barbules shown in Fig. 3 [11]. On the other hand, external forces can break the interlocking if the total external force exceeds that exerted by the little hooks. To simulate this effect, we introduce external forces into our parametric L-system as follows

$$\begin{array}{l} \omega : R(0,0,0) \\ p_1 : R(i,F_L,F_R) : i < N \;\&\&\; F_L \le F_0 \;\&\&\; F_R \le F_0 \\ \quad \rightarrow [B_L(i,0)][B_R(i,0)]R(i+1, F_L+F_e, F_R+F_e) \\ p_2 : R(i,F_L,F_R) : i < N \;\&\&\; F_L > F_0 \;\&\&\; F_R \le F_0 \\ \quad \rightarrow [O_L B_L(i,0)][B_R(i,0)]R(i+1, 0, F_R+F_e) \\ p_3 : R(i,F_L,F_R) : i < N \;\&\&\; F_L \le F_0 \;\&\&\; F_R > F_0 \\ \quad \rightarrow [B_L(i,0)][O_R B_R(i,0)]R(i+1, F_L+F_e, 0) \\ p_4 : R(i,F_L,F_R) : i < N \;\&\&\; F_L > F_0 \;\&\&\; F_R > F_0 \\ \quad \rightarrow [O_L B_L(i,0)][O_R B_R(i,0)]R(i+1, 0, 0) \\ p_5 : B_L(i,j) : j < M_L \rightarrow B_L(i,j+1) \\ p_6 : B_R(i,j) : j < M_R \rightarrow B_R(i,j+1) \end{array}$$

where F_L and F_R are the total external forces on the left and right barbs respectively. O_L and O_R are directional rotations of the left and right barbs in response to F_L and F_R. The productions p_1 through p_4 basically say that for each step we move along the rachis curve, we increment F_L and F_R by a random external force F_e. If at some point F_L (F_R) exceeds the force F_0 exerted by the cilium, the left (right) barb is rotated by a random angle θ in a direction determined by F_L (F_R). The rotation of a barb B is assumed to be within the tangent plane defined by the tangent vectors of the rachis and barb B at the point where the rachis and barb B intersect. The random rotation angle is computed as $\theta = \lambda r_k(s)$, where $r_k(s)$ is the k-th random number generated with the random seed s and λ is a user-defined constant. After the rotation, F_L (F_R) starts to accumulate again from zero. The random seed of each feather is saved so that its shape remains the same every time it is rendered. Fig. 4 contains a number of feathers created with random gaps between the barbs. Fig. 4 also exhibits feathers of different types and shapes by changing parameters of the L-system.

Figure 4: The leftmost feather is generated without random gaps in the feather blade. The other four feathers demonstrate different kinds of feathers that can be generated by changing a few parameters in the feather modeling system.

3.2 Feather Appearance

We use a BTF to capture the mesostructure and the directional radiance distribution at each point on the feather surface. A BTF is a 6D function $T(x, y, \theta_v, \phi_v, \theta_l, \phi_l)$, where (θ_v, ϕ_v) is the viewing direction $\mathbf{v}$ and (θ_l, ϕ_l) is the lighting direction $\mathbf{l}$ at surface point (x, y) [3]. To calculate the BTF, we built a geometry model for the

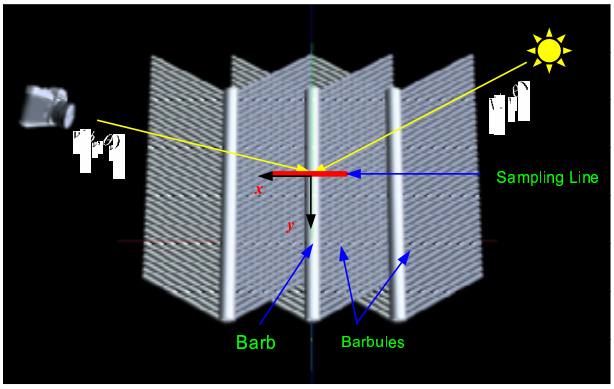

Figure 5: Sampling the BTF on the barb-barbules mesostructure. The sampling is done along the horizontal line highlighted in red.

barbs and barbules as shown in Fig. 5 and render this structure for all viewing and lighting settings. Since the rendering is done off-line, we can afford complicated geometry and sophisticated lighting models. The geometry shown in Fig. 5 is built according to the anatomy of the barb-barbule structure. This model is opaque with both diffuse and specular reflections.

As Fig. 5 indicates, we only sample the BTF along the x-axis to obtain a 5D BTF

$$T_{bb}(x, \theta_v, \phi_v, \theta_l, \phi_l) = T(x, y_0, \theta_v, \phi_v, \theta_l, \phi_l)$$

for some constant y_0. As we shall see, this 5D BTF T_{bb} suffices for rendering the actual 6D BTF of a feather because of the spatial arrangement of barbs and barbules. We render the mesostructure of barbs and barbules such that fine-scale shadows, occlusions, and specularities are well-captured in T_{bb}. The rendering is done off-line by a ray-tracer.

The above model of feather mesostructure has a number of advantages. First, the off-line BTF calculations allow us to capture a very complicated mesostructure and directional radiance distribution at each surface point. Second, the BTF can model additional effects such as oil-film interference and iridescence, which is important for a class of familiar birds such as hummingbirds and ducks [19]. Finally, we can easily support level-of-detail rendering with a BTF. In close-up views, a BTF shows the mesostructure, as Fig. 7 demonstrates. As the viewing distant increases, we can mipmap the BTF, which eventually becomes a BRDF at a distance.

3.3 Feather Rendering

When rendering a feather, we call our parametric L-system to generate the feather at run-time. The storage requirement is modest for each feather because only its L-system parameters and the random seeds are stored; details such as barb curves (polylines) and the random gaps on the vane (the feather blade). As illustrated in Fig. 3, a feather is composed of a series of barbs on both side of the rachis and each barb has its barbules. We use the pre-computed 5D BTF T_{bb} to efficiently draw the barb-barbule mesostructure and thus achieve realistic rendering for a wide range of viewing distances.

Fig. 6 illustrates the rendering of a feather. The feather L-system describes a barb B by a polyline L_B with vertices $\{\mathbf{x}_0, ..., \mathbf{x}_n\}$. When we render a barb, we want to render the barb B as well as the barbules attached to B. For this reason, we build a quadrilateral strip along the polyline L_B as shown in Fig. 6. The local lighting direction $\mathbf{l}(\mathbf{x}_i)$ and viewing direction $\mathbf{v}(\mathbf{x}_i)$ are calculated at every vertex $\mathbf{x}_i$ of L_B using the local coordinate frame at $\mathbf{x}_i$. On each short edge E across the barb polyline L_B, a 1D texture is

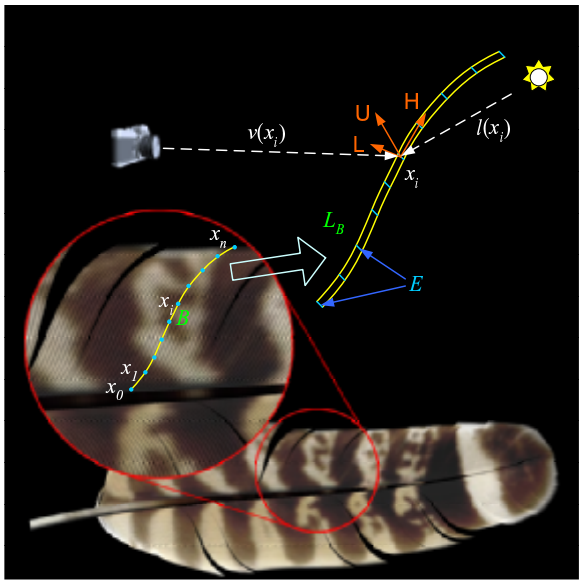

Figure 6: Feather rendering. Each barb curve is drawn as a quadrilateral strip using the pre-computed 5D BTF.

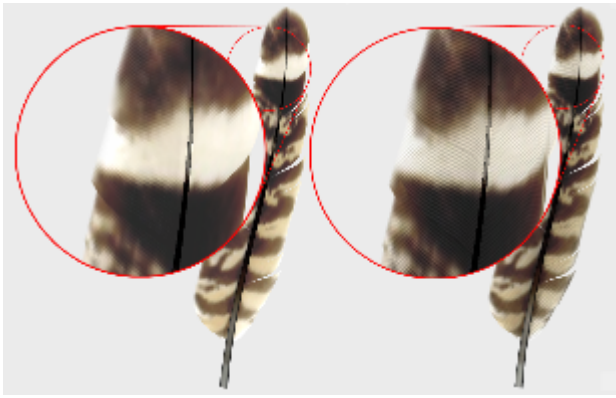

Figure 7: Left: A feather rendered without the BTF. Right: A feather rendered with the BTF.

created by looking up color values from T_{bb} using the directions $\mathbf{v}(\mathbf{x}_i)$ and $\mathbf{l}(\mathbf{x}_i)$. Thus we obtain $(n+1)$ 1D textures of resolution n_b where n_b is the spatial resolution of the BTF T_{bb}. These textures are combined with the RGBA texture of the feather (see window (3) of Fig. 2) to render the barb B by multi-texturing and alpha-blending using graphics hardware. Fig. 7 compares feather rendering with and without the BTF.

Fig. 8 illustrates different effects that can be achieved with the BTF. When sampling the BTF with a ray tracer, we can adjust parameters so that the BTF gives a "hard" or "soft" appearance to the feather. Fig. 8 (c) and (d) demonstrate occlusions and specularities caused by barb mesostructure.

4 Feathering a Bird

4.1 Wings and Tail

We first construct feathers on the wings and the tail using skeletons. The feathers on a wing include the primaries, secondaries, humerals, primary coverts, and secondary coverts (these feathers

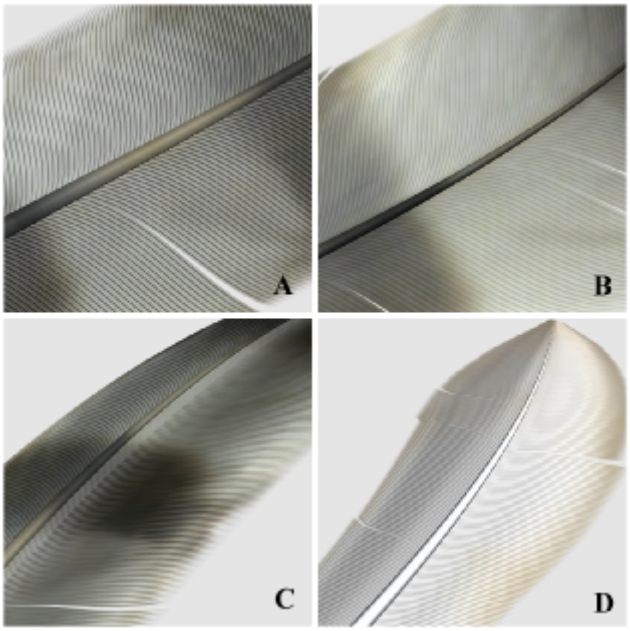

Figure 8: (a) A feather with a "hard" appearance. (b) A feather with a "soft" appearance. (c) Local occlusion on the feather blade. (d) Local specularities on the feather blade.

are rooted on the scapula, ulna/radius, metacarpus and phalanx respectively) [18]. As shown in Fig. 9, we use a polyline consisting of four line segments to represent the wing skeleton. Similarly a quadrilateral is used as the skeleton for the tail. Generally speaking, a bird has about 9 to 11 primaries, 6 to 24 secondaries, and 8 to 24 tail feathers [11, 18]. In our system, the user specifies the numbers of feathers of each type and edits 8 key feathers on the wing and 4 key feathers on the tail. The system generates other feathers by interpolation. Fig. 9 illustrates the feather placement on a wing.

4.2 Contour Feathers

Contour feathers are the feathers that cover the body of a bird. Given a polygonal model describing a bird's body (without feathers), we want to place feathers of different sizes and shapes on the model. The huge number of feathers on a bird makes it impossible to manually place and edit individual feathers. In our system, we let the user specify a number of key feathers and their growing directions; the system automatically generates a full coverage of the bird based on the key feathers. This full coverage is created in three steps:

a) re-tile the polygonal model to generate feather growing positions,
b) interpolate the key feather growing directions to all feather growing points to get an initial growing direction at each point, and
c) recursively determine the final feather growing direction at every feather growing point, with collisions between feathers detected and rectified.

The output is a feather placement map indicating feather growing positions and directions. The feather shape parameters are interpolated from that of nearby key feathers. These shape parameters are used to generate a simplified geometry for each feather. This simplified geometry is used for collision detection in step (c).

For an interpolated feather, the random gaps on the feather blade is generated by giving the feather a new random seed.

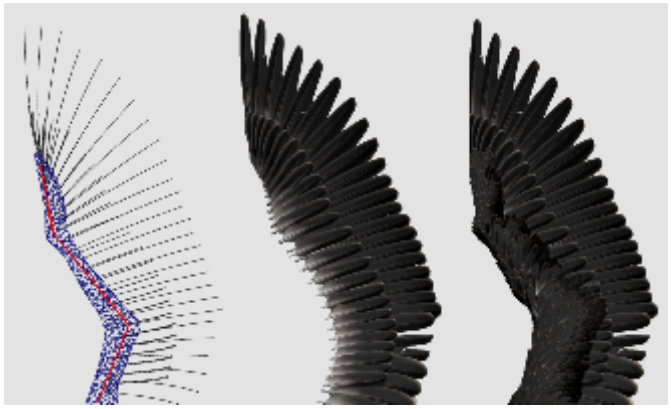

Figure 9: Feather placement on a wing. Left: The skeleton is drawn in red over the geometry model of the wing. The black lines show the position and orientation of flight feathers. Middle: Rendering of flight feathers. Right: Rendering of flight feathers along with other small wing feathers.

Feather Growing Positions: The vertices of the given polygonal model usually cannot be used directly as feather growing positions. The feathers at different parts of a bird have different sizes, and small feathers need to grow densely in order to cover the bird's skin. In addition, feathers tend to distribute evenly in a region of constant feather density. Vertices of a polygonal model often do not have these properties.

To address this problem, we retile the polygonal model using Turk's algorithm [21]. This retiling creates a polygonal model whose vertices are evenly distributed over a region of constant density. Turk also introduced a simple technique for adjusting vertex density based on curvature. We adapt this technique to controlling vertex density based on the sizes of feathers, which are interpolated from that of the key feathers using Gaussian radial basis functions, where radius is defined as distance over the mesh, as computed using Dijkstra's shortest path algorithm. The user has control over the spatial extent and weight of each basis function. This interpolation scheme is similar to that used by Praun et al. for creating vector fields on a polygonal surface [12].

After retiling, the vertices of the new polygonal model are the feather growing positions.

Feather Growing Directions: From the growing directions of key feathers, we calculate initial growing directions at all vertices using Gaussian radial basis functions as described before. The initial growing directions tend to cause inter-penetration of feathers because these directions are derived without any consideration to the shapes of the feathers. To determine the final growing directions, we need to perform collision detection on the feathers based on their simplified geometry and to adjust the feather growing directions accordingly. Because of the large number of feathers and the complex shape of a bird's body, a collision detection between every pair of feathers is likely to be very expensive. To address this problem we adopt two strategies. First, we grow feathers in an orderly fashion according to the initial growing directions. Second, we only consider local collisions between neighboring feathers because collisions rarely happen between feathers far away from each other (two feathers are neighboring feathers if their growing positions are connected by an edge). We implement these two strategies using a recursive collision detection algorithm.

As Fig. 10 illustrates, we first classify the vertices around each vertex $\mathbf{v}$ into two groups according to the initial growing direction $\mathbf{o}_v$ at $\mathbf{v}$. The first group consists of vertices $\{\mathbf{v}' \mid \mathbf{v}' \prec \mathbf{v}\}$, where $\mathbf{v}' \prec \mathbf{v}$ means $(\mathbf{v}' - \mathbf{v}) \cdot \mathbf{o}_v > 0$. The second group is

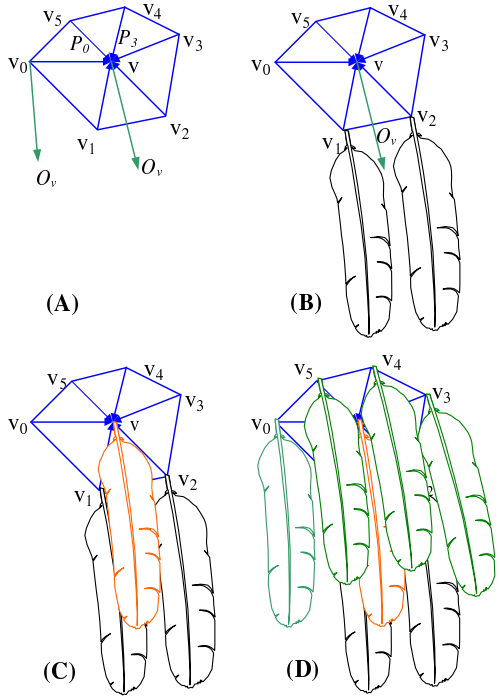

Figure 10: Recursive collision for eliminating the inter-penetration of neighboring feathers.

$\{\mathbf{v}' \mid \mathbf{v}' \succ \mathbf{v}\}$, with $\mathbf{v}' \succ \mathbf{v}$ meaning $(\mathbf{v}' - \mathbf{v}) \cdot \mathbf{o}_v \leq 0$. After the vertices around every vertex are so classified, we invoke the following recursive collision detection algorithm at every vertex.

```
FindGrowingDirection(v)
{
  If the growing direction at v has already been found
    return;
  For each vertex v' ≺ v
    FindGrowingDirection(v');
  While feather(v) collides with feather(v') for some v' ≺ v
    adjust the growing direction at v;
  For each vertex v' ≻ v
    FindGrowingDirection(v');
}
```

Here feather($\mathbf{v}$) and feather($\mathbf{v}'$) are the feathers at vertices $\mathbf{v}$ and $\mathbf{v}'$ respectively.

To see how this algorithm works, consider a vertex $\mathbf{v}$ on the retiled model with initial growing direction $\mathbf{o}_v$, as is shown in Fig 10. The vertices around $\mathbf{v}$ are vertices $\mathbf{v}_0$ through $\mathbf{v}_5$. Among these, $\mathbf{v}_1 \prec \mathbf{v}$ and $\mathbf{v}_2 \prec \mathbf{v}$, whereas $\mathbf{v}_0 \succ \mathbf{v}$, $\mathbf{v}_3 \succ \mathbf{v}$, $\mathbf{v}_4 \succ \mathbf{v}$ and $\mathbf{v}_5 \succ \mathbf{v}$. The algorithm will first determine the final growing directions at $\mathbf{v}_1$ and $\mathbf{v}_2$ by recursion. Based on the final growing directions at $\mathbf{v}_1$ and $\mathbf{v}_2$ as well as the shapes of feathers at $\mathbf{v}$, $\mathbf{v}_1$, and $\mathbf{v}_2$, we can detect the collision between these feathers and adjust the feather growing direction at $\mathbf{v}$ by rotating it toward the surface normal at $\mathbf{v}$. We rotate in small increments, stopping as soon as no more collisions are detected. The growing direction at $\mathbf{v}$ is now considered final and we can process $\mathbf{v}_0$, $\mathbf{v}_3$, $\mathbf{v}_4$, and $\mathbf{v}_5$ through recursions. For fast collision detection, a simplified geometry is used for each feather. Fig. 11 shows the results of the final growing directions.

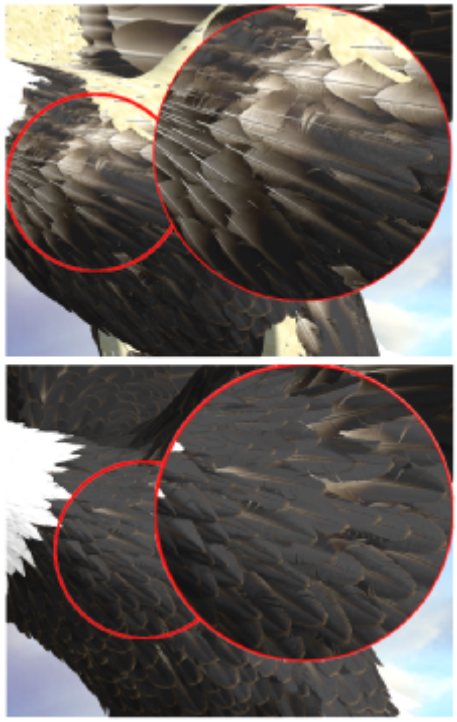

Figure 11: The effect of recursive collision detection. Top: Feathers rendered according to the initial growing directions. We see many inter-penetrating feathers. Many feathers also grow into inside the bird's skin (rendered as a yellow surface). Bottom: Feathers rendered according to their final growing directions.

5 Results

We have implemented our feather modeling and rendering system on a PC with a 864 MHz Pentium III processor, 256M RAM and an NVIDIA GeForce3 display adaptor.

Fig. 12 shows two images. On the top is a real image of a feather, while a rendering from a similar viewing distance is shown on the bottom. Comparison of the two images demonstrates the similarity of our modeling and rendering result to an actual feather. Notice the mesostructure and random gaps on the feather blade are realistically portrayed in the synthetic feather. These effects would be very difficult to capture using textured polygons. It usually takes the user a few minutes to model a feather.

Fig. 13 shows an American Indian Headdress. The feathers in the image are modeled and rendered using our system. The feathers were manually placed by a graphics artist. The white plume-like effect is not rendered by our system but added by postprocessing.

Fig. 1 shows an eagle modeled and rendered using our system. About 3500 feathers were placed on the bird. It took the user about 30 minutes to specify the 50 key feathers. Determining the feather growing positions and the initial growing directions took a few minutes. The most time-consuming step is the calculation of the final growing directions, which took about 30 minutes. Rendering speed is about one minute per frame.

6 Conclusions

We have presented techniques for modeling and rendering feathers. Our main idea is to represent a feather as a branching structure described by a parametric L-system. This approach allows the user to

Figure 12: Top: Photograph of a real feather. Bottom: A synthetic feather modeled and rendered using our system.

Figure 13: All feathers in this American Indian headdress are modeled and rendered using our system.

generate feathers of different types and shapes by adjusting a few parameters. The randomness in feather geometry is also simulated in the parametric L-system we derived. We described a rendering algorithm based on this L-system and an efficient form of the BTF representing the mesostructure of a feather blade. This algorithm can efficiently generate photo-realistic renderings of feathers. We also developed a system that allows the user to easily create a large number of feathers on a bird's body and render the bird realistically.

It is our hope that this work will stimulate other researchers to explore the beautiful world of birds. Our work is only a first step towards realistic and efficient rendering of feathers and birds. Many aspects of this topic need further research. A limitation of our current system is that it does not support down feather rendering. Another limitation is that we do not handle oil-film interference and iridescence [19], which is important for a class of familiar birds such as ducks and hummingbirds. We plan to address these issues in the near future. Another interesting area of future work is bird animation [16]. The flocking of birds has been explored by Reynolds [17]. More work is needed to understand how individual birds fly. Finally, we are interested in applying L-systems to other natural phenomena.

Acknowledgments

We would like to thank Xinguo Liu for useful discussions. Many thanks to Dongyu Cao for her illustrations, to Yin Li, Gang Chen, and Steve Lin for their help in video production, to Steve Lin for proofreading this paper, and to anonymous reviewers for their constructive critique, which has significantly improved the presentation of this paper.

References

[1] Wen-Kai Dai, Zen-Chung Shih, and Ruei-Chuan Chang. Synthesizing feather textures in galliformes. *Computer Graphics Forum*, 14(3):407–420, August 1995.

[2] Kristin J. Dana and Shree Nayar. 3d textured surface modeling. In *Proceedings of IEEE Workshop on the Integration of Appearance and Geometric Methods in Object Recognition*, pages 46–56, June 1999.

[3] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[4] Cristiano G. Franco and Marcelo Walter. Modeling the structure of feathers. In *Proceedings of SIBGRAPI 2001 - XIV Brazilian Symposium on Computer Graphics and Image Processing*, page 381, October 2001.

[5] James Hanan. *Parametric L-Systems and Their Application to the Modeling and Visualization of Plants*. PhD Thesis, University of Regina, 1992.

[6] Jan J. Koenderink and Andrea J. Van Doorn. Illuminance texture due to surface mesostructure. *Journal of the Optical Society of America*, 13(3):452–463, 1996.

[7] Aristid Lindenmayer. Mathematical models for cellular interaction in development, parts i and ii. *Journal of Theoretical Biology*, 18:280–315, 1968.

[8] Xinguo Liu, Yizhou Yu, and Heung-Yeung Shum. Synthesizing bidirectional texture functions for real-world surfaces. Computer Graphics Proceedings, Annual Conference Series, pages 97–106, August 2001.

[9] Tom Malzbender, Dan Gelb, and Hans Wolters. Polynomial texture maps. *Proceedings of SIGGRAPH 2001*, pages 519–528, August 2001.

[10] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 301–308, August 2001.

[11] Christopher M. Perrins and Alex L. A. Middleton. *The Encyclopedia of Birds*. Checkmark Books, 1985.

[12] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. *Proceedings of SIGGRAPH 2000*, pages 465–470, July 2000.

[13] Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomír Mech. Visual models of plant development. *Handbook of Formal Languages*, 1996.

[14] Przemyslaw Prusinkiewicz, Mark Hammel, Radomír Mech, and Jim Hanan. The artificial life of plants. *SIGGRAPH 95 Course Notes*, 7:1–38, 1995.

[15] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[16] Balajee Ramakrishnananda and Kok Cheong Wong. Animating bird flight using aerodynamics. *The Visual Computer*, 15(10):494–508, 1999.

[17] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, volume 21, pages 25–34, Anaheim, California, July 1987.

[18] Bart Rulon. *Painting Birds Step by Step*. North Light Books, 1996.

[19] Morgan Schramm, Jay Gondek, and Gary Meyer. Light scattering simulations using complex subsurface models. In *Graphics Interface '97*, pages 56–67, May 1997.

[20] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. Computer Graphics Proceedings, Annual Conference Series, July 2002.

[21] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):55–64, July 1992.