# Building Efficient Fur Pipeline for a Low Cost Production of Creature-based Feature Film

Wanho Choi [*]
Dexter Digital
Seoul National University

Taekyung Yoo [†]
Dexter Digital

Sanghun Kim [‡]
Dexter Digital

Hyeong-Seok Ko [§]
Seoul National University

Tae-Yong Kim [¶]
nVidia

## Abstract

Despite impressive progresses for creating furry creatures in feature films, building a practical fur pipeline for digital creatures remains a major technical challenge for feature films. A few commercial packages exist, but they still lack in ability to fully support production of hundreds or thousands of shots required for a creature-based feature film, both economically and technically. Thus, only a few elite production houses can afford to build a proprietary hair pipeline dedicated for production of such movies, and anything involving large amount of fur generation/rendering remains beyond the budget of smaller, low cost productions. In this talk, we present our experiences in building an efficient, practical fur pipeline for generating 900+ shots containing two digital gorillas for the production of the movie *Mr. Go*, a mid cost VFX film totaling about $10M for the entire VFX budget. We provide details of our creature pipeline, and describe lessons we learned in our attempts to leverage both commercial and inhouse software tools to meet the quality requirement while staying as competitive as possible in the budget. We also discuss how both commercial software and future research in this area could improve construction of such pipeline in the future.

## 1 Introduction

Despite recent successes in high profile movies such as *King Kong* and *Avatar*, creating believable digital creatures remains one of the biggest challenges facing current digital production studios, both economically and technically. Among such challenges, creating the appearance of furry creatures is one of the most expensive options, thus only a few elite studios who have developed their respective fur pipeline over many years are capable of producing such effects. This limits wider adoption of creature-based stories even though demands for such work are increasingly becoming stronger. Especially, many shallow pocketed film makers often find themselves frustrated after finding that creating such a creature-based film remains still very expensive options for them, due to limited availability of technology and expertise.

A few commercial software packages do exist for creating fur and hair for digital films, but they still lack in ability to support production-level requirements that arise from massive production of hundreds or even thousands of shots involving fur. Such tools often offer an excellent workflow for a single or small group of users, suited for a small scale production. But when these tools are put into a large, production scale, they do not offer flexibility required to adapt to individual shot, nor scale well to massive rendering of many shots on render farms.

For the production of upcoming movie *Mr. Go*, the authors went through a painful process of transitioning from naive user of off-the-shelve tools to building a complete production fur pipeline to meet both quality and budget requirement of this production, a mid scale feature film totaling about $10M in VFX budget, including a full stereo pipeline for 900+ shots involving two digital gorillas. The budget was miniscule compared to typical Hollywood production budget for such creatures. So we had to learn how to leverage existing off-the-shelf technology to meet such production requirements. We believe that many other smaller VFX shops and film makers would go through similar endeavours, and hope our presentation here could shed some lights on how to create efficient, budget oriented pipeline for furry creatures for small to mid scale feature film productions. Although we do not intend to claim that the presented pipeline is novel, we do have some novel twists we made to make the pipeline more affordable and practical, hence we want to share those in this talk.

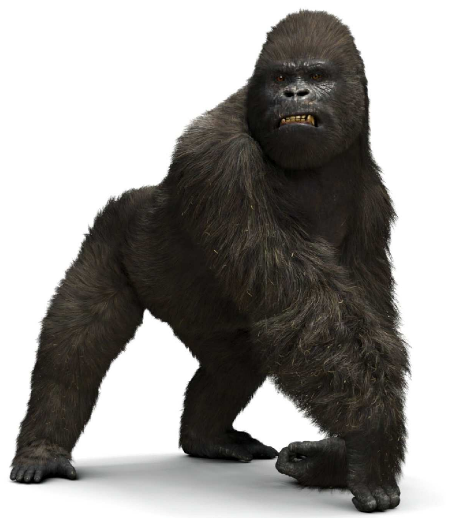

Figure 1: A gorilla character of which fur strands were generated with our methods. Finally, 979,526 full fur strands were generated from 64,920 triangles and 13,030 guide strands. It ran at about 7 fps for the guide strands simulation only, and about 2 fps including the full fur interpolation.

## 2 Practical hair rendering pipeline for a film production

This movie contains about 900 shots involving two digital gorilla characters, so it became very clear from the early stage that hair rendering would be the dominant production cost, from render farm, to rendering software cost, and to production labor cost. So the efforts presented here are mostly focused on minimizing the following costs.

- The total render time for fur. This not only includes shading and actual renderer time, but also includes fur generation/import time.
- Fur generation and render time scales with number of ren-

[*]e-mail: fxwano@gmail.com
[†]e-mail: utd.vfx@gmail.com
[‡]e-mail: masin@gmail.com
[§]e-mail: hsko@graphics.snu.ac.kr
[¶]e-mail: hsko@graphics.snu.ac.kr

dered hairs, so reducing number of rendered hair without sacrificing visual fidelity is an important goal.

- Total production cost depends on how many iterations artists need at each step of the pipeline, including grooming, animation, simulation and lighting. Reducing number of per-shot iteration was another important goal.

### 2.1 Cache file size optimization

There exist a few commercial packages that specialize in grooming, rendering, and simulating hair and fur. One of the most notable and popular example is the *Shave and a Haircut* software. We initially investigated this tool and gradually moved away from using this in the pipeline, so most discussions here follow our experience with this software. But the discussions would equally apply to other commercial packages.

Commercial packages provide in-place render solutions through their host software such as Maya, but this makes integration with external renderer complex and less flexible than desired by production pipeline. For example, many shots require specialized shaders that are not easily integrated into such pipeline, and the production team felt that the quality of rendered images were not as good as dedicated rendering solutions such as *RenderMan*. So, it was necessary to break the hair pipeline into grooming stage where any interactive hair modeling package could be used, but have separate render-time solutions. A natural choice would be to export all the hairs from the grooming software, and then render those in a separate offline render batch. However, this process puts a significant pressure on both disk I/O bandwidth as well as storage requirement. For example, just storing one frame of hair data for 700K hairs would consume about 256Mb in file size. This would translate to 6Gb of file data for one second worth of frames, 360Gb for one minute of frames, and 20Tb for the full hour of movie frames! This not only makes the export process slow, but also creates an enormous pressure on disk storage system. On the other hand, exporting only the guide hairs and procedurally generating final render hair in render-time using DSO cuts such bandwidth by two orders of magnitude (see Figure 3). In this approach, each frame of hair data could be stored in 5.4Mb, about 50x savings in both bandwidth and disk storage. This results in improved render time (Figure 4), as well as additional flexibility in choice of grooming parameters.

### 2.2 Implementing Render-time DSO for fur generation

Methods to generate interpolated hairs from guide hair are well known, so we only sketch our system briefly here. Given the character's model geometry which was shared across modeling, animation, and rendering stages, the system imports guide hairs and density maps. The root position of every hair is sampled based on

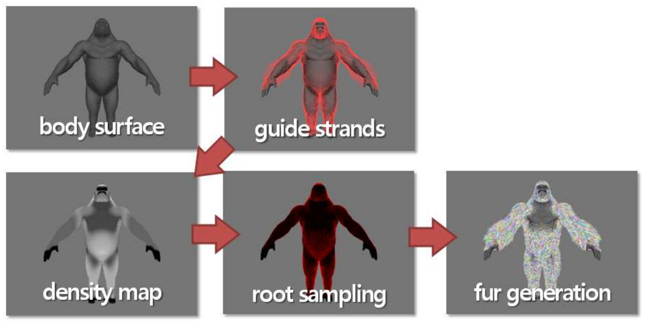

Figure 2: Overview of fur generation process in our fur system.

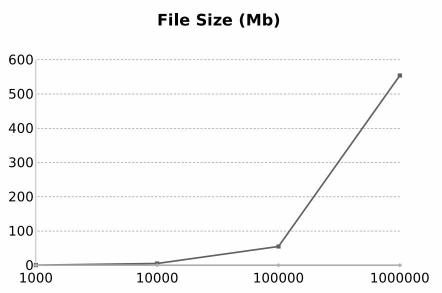

Figure 3: A comparison of file size between fur imported from exported RIB file vs fur generated with DSO on rendering time.

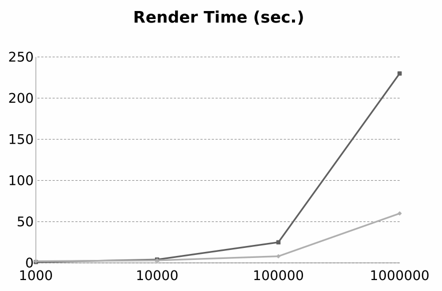

Figure 4: A comparison of render time between fur imported from exported RIB file vs fur generated with DSO on rendering time.

the density, and stored along with the guide hair data (Figure 2). The render-time DSO then generates every rendered hair, using a set of additional texture maps to vary grooming parameters such as bending, kink, clumping, etc.

For interpolating the dense full fur strands from the sparse guide strands, we use the similar approach as [**?**]. To interpolate positions and other attributes of dense hairs from guide strands, we first find a number of neighboring guide strands of the fur strand. In this step, we exclude any guide strands having reverse surface normals at their roots compared to the surface normal of the fur strand. To reduce the interpolation artifacts, we randomize the number of neighbor guide strands per each fur strand. The control points of the neighboring guides are first transformed to the local space of the strand, and these points are averaged with harmonic weights, which is inversely proportional to the distance between the roots in the world coordinates. To add kinkle, we add small perturbations to the points. Finally, the resulting points are transformed back to the world coordinate.

### 2.3 Optimizing Hair Counts by Poisson Sampling

If fur strands are not well distributed over the body mesh, the skin underneath them can be visible especially when they are rendered at a close distance. It is known that the distribution of fur strands typically follow a Poisson disk pattern [**?**], and we observed that such sampling patterns significantly improves the coverage of hairs, thus allowing us to use smaller number of hairs for the desired visual look. For final look, we could reduce number of hairs down to less

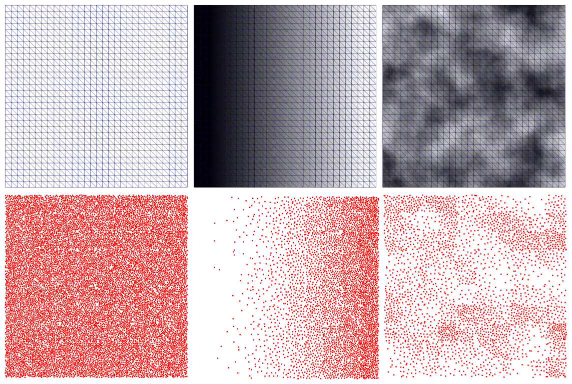
Figure 5: Controlling the sampling density. The gray scale image maps (top row), and the samples generated from them (bottom row). 11,175 (left column), 3,992 (middle column), and 2,909 (right column) points were generated in 0.06∼0.07 sec., 0.05∼0.06 sec., and 0.06∼0.07 sec., respectively.

than one million, from 3-4 million hairs we started with random sampling. Any sampling methods that would give Poisson patterns on a given surface would do the job here, including [?], [?] and [?].

We developed a simple, yet efficient sampling method, which we call the Poisson stamping method, to sample evenly spaced root positions from each triangle of the body mesh. Firstly, we perform 2D Poisson disk sampling for a rectangle, which we call the Poisson stamp, using the technique proposed by Bridson [?] with $r = 0.35/(\sqrt{N_{target} \times A_{normalized\_max}})$ where $N_{target}$ is the desired number of fur strands and $A_{normalized\_max} = max(A_1/\sum A_j, \ldots, A_m/\sum A_j)$ with $A_j$ is the are of the $j$-th triangle. The size of the rectangle is determined so that any mesh triangle can be placed properly inside of it after applying an arbitrary 2D rotation to the triangle. To determine the root positions for each body mesh triangle, we place the triangle at a random location within the Poisson stamp, and then the samples from the stamp are copied to the triangle. Each sample stores barycentric coordiates so that they can be used to restore root positions in 3D.

The Poisson stamping method is easy to implement and works very fast since it has only to check whether a point is inside a triangle in 2D. Moreover, the method allows to control the sampling density. When a triangle is painted in gray scale by artists, it would be down scaled by half size before stamping while no scaling for the white region, and the degree of down scaling increases as the region gets darker. The sampling density can be controlled as shown in Figure 5.

In a strict sense, our method cannot generate a perfect Poisson disk distribution since the points at the vicinity of triangle edges can violate the Poisson disk sampling properties. However, the method produces quite plausible results. All experiments performed with the Poisson stamping method did not exhibit any conspicuous artifacts.

## 3 Dynamic Simulations

We follow the standard approach of only simulating guide hairs, and using interpolation for the rest of hairs. For simulations, we used a three-level hierarchy in hairs, a top level guide hairs from grooming, mid level guide hairs for dynamics simulation, and the dense hair used for rendering. Thus, a typical iteration of a simulation pass involves running dynamics on both top guide hairs and mid hairs, interpolating these into full render hairs, running rendering for verification of results.

Note that certain issues such as hair penetration only reveals when hair is fully lit with shadows, etc., and increased number of simulation pass often means increased render time in production. In light of lowering render costs, it is desirable to have a robust simulation approach that can withstand even physically implausible character motions - we found that most of existing off the shelf hair simulation tools do not always offer such behavior. A simulation failure for a single guide hair often means many repeated iteration of such simulation to fix the issue, or painful manual intervention by artists.

### 3.1 Simplified Strand Dynamics with Mass Spring and CCD IK

In addition to the conventional dynamics toolchains, we added a pseudo dynamics system where we only simulate the tip of each hair strand, attached to the corresponding point at the rest pose by a spring mass system. Given this oscillatory motion of the tip point, each guide strand is kinematically animated as an articulated rigid body by regarding its control points as rotating joints and the tip as an end-effector. At each time step, we perform the following steps for each guide strand: (1) Find the neutral configuration. (2) Run the mass-spring system to update the tip position, in which the tip is connected to the tip of that neutral configuration via a spring with zero rest length. (3) Determine the joint angles by applying inverse kinematics to the updated tip position. (4) Generate a Catmull-Rom spline which passes the joints. The Catmull-Rom spline is used for the visualization and the interpolation for the full fur strands. Figure **??** shows these steps.

The mass spring simulation of the tip point provided three control parameters for frequency, amplitude, and decay of each spring and these could vary by positions and user provided maps. Once the position of the tip is determined, the rest of hair is updated using the Cyclic Coordinate Descent (CCD) method [?] as follows. It adjusts the joint angles from tip to root in serial order so that the end-effector heads toward the given target position as shown in Figure 7. This process is repeated until the distance between the end-effector and the target position is within the tolerance or the number of iterations reaches the user specified maximum. When the end-effector cannot reach the target position, CCD IK solver naturally imposes a feasible end-effector position to be as close as possible to the target position.

This simulation process provided a very intuitive control for users while the resulting simulations always remain stable. Our simple dynamics model worked well usually, but sometimes we needed more robust dynamics model for resolving accurate collisions or generating high-speed motions. When higher realism was needed, we would resort back to the existing simulators (e.g. *Maya nHair*) to re-simulate only those regions from the cached simulation data.

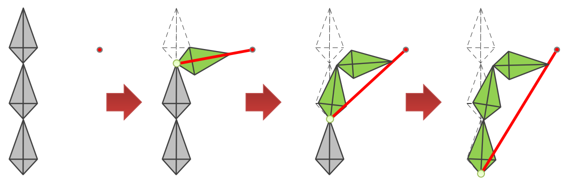
Figure 7: Working of CCD IK for the rigid articulate body consisting of three links. This figure shows only the first three steps of the whole procedure.

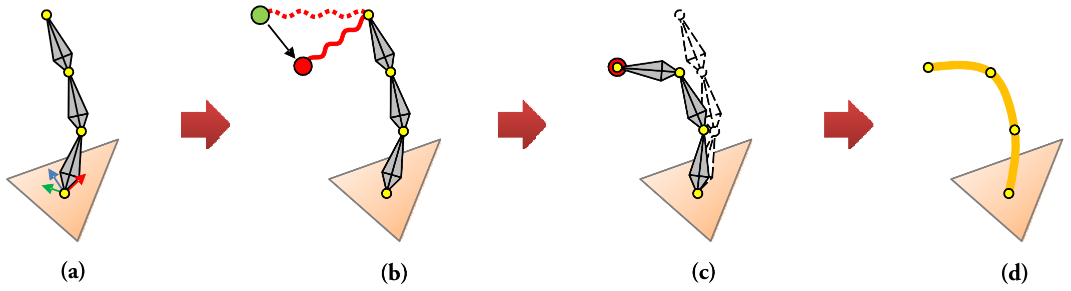

Figure 6: Tip position update with the mass-spring system. (a) The neutral configuration. (b) The green circle represents the tip at the previous time step. The red circle represents the updated tip position with the mass-spring system. (c) The result of inverse kinematics applied to the updated tip. (d) The Catmull-Rom spline for the visualization.

### 3.2 Strand-to-Strand Interactions

Strands interact with each other due to collisions and frictions. Taking advantage of the simplicity the tip-point dynamics has, we also resolve strand-to-strand interactions at the tip-point level. At the initial state, we put links between the tip points of nearby guide strands to enforce lateral constraints. Since dynamic effects are already considered in the stage of the mass-spring simulation, we apply only a few iterations of relaxation on this network.

### 3.3 Dealing with Collisions and Penetrations

The guide strands may penetrate the objects or body itself, in which case we apply a rigid rotation about the root so that the deepest point comes out of the surface. Interpolated hairs, however, may still penetrate even when the guide strands are collision-free. In such cases, we cascade the collision avoidance processing to the dense render hair for regions specified by the user.

On areas where hairs are sandwitched between two layers (e.g. the body and the cloth), hairs would penetrate on the boundary of such objects even after the collision avoidance processing. For such cases, we detected intersection of each guide hair against the collision geometry, and the information such as hair id and first control vertex of the colliding hairs was recorded and passed to the render time hair generator. At the runtime hair generation stage, each interpolated hair would be trimmed based on that info. When hairs are trimmed independently per each frame, hair length can change and same hairs may pop in and out when they are near boundary of the collision objects. To avoid this disturbing artifact, we used a global post processing tool that scans entire frame range to determine the shortest length for each trimmed hair, and apply the length for trimming those hairs for the entire shot.

## 4 Lighting and Other Render Optimization

To allow efficient composite time tuning, lighting produced multiple render passes, such as ambient occlusion, diffuse and specular, etc. To avoid excessive renders, a few representative frames were first sampled and then the full frames were sent to render farm for final rendering. We also discovered that using smaller number of hairs for shadow map generation did not significantly degrade visual quality, and used about half of hairs for shadow map generation. Figure **??** shows each render pass and the final render of an example shot. Our average rendering time was 30$\sim$40 minutes, and we took about ten times for obtaining final image in the rendering stage.

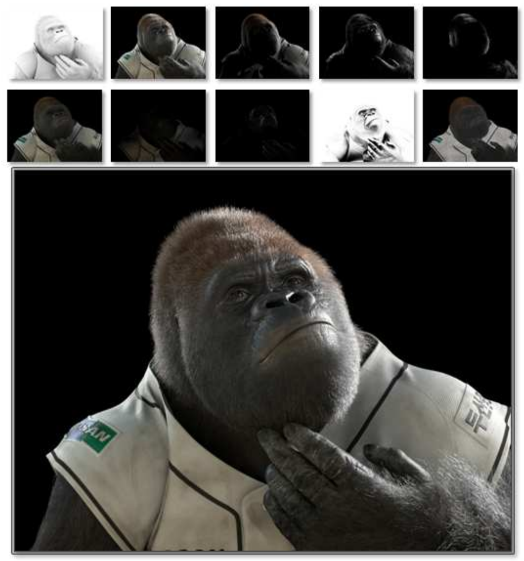

Figure 8: Multiple render passes. The final render image is shown at the bottom.

## 5 Conclusions

Fur/hair pipeline is often a very complex system, where a blend of different technologies have to work together. We have presented our experiences building a feature level hair pipeline based on a mix of commercial packages and our own set of techniques. Most current off-the shelf tools have strength and weakness in each specific area, but they often lack in ability to blend seamlessly with other tools. From our experience, dealing with fur pipeline often requires us to add new set of features that may not be present in existing tools. So providing more flexibility and interoperability would be one important avenue for enhancing useability of existing and new hair packages.

## 6 Acknowledgement

This work was supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Culture Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Programs 2013, the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (MEST) (No. 2012R1A2A1A01004891), the Brain Korea 21 Project in 2013, and ASRI (Automation and Systems Research Institute at Seoul National University).

## References

BOWERS, J., WANG, R., WEI, L.-Y., AND MALETZ, D. 2010. Parallel poisson disk sampling with spectrum analysis on surfaces. In *ACM SIGGRAPH Asia 2010 papers*, 166:1–166:10.

BRIDSON, R. 2007. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 sketches*, SIGGRAPH '07.

CHANG, J. T., JIN, J., AND YU, Y. 2002. A practical model for hair mutual interactions. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, 73–80.

CLINE, D., JESCHKE, S., RAZDAN, A., WHITE, K., AND WONKA, P. 2009. Dart throwing on surfaces. *Comput. Graph. Forum 28*, 4 (June), 1217–1226.

GOLDMAN, D. B. 1997. Fake fur rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, 127–134.

TURK, G. 1992. Re-tiling polygonal surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, 55–64.

WANG, L. C. T., AND CHEN, C. C. 1991. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans. Robotics and Automation 7*, 4, 489–499.