# Non-Photorealistic Rendering of Hair for Animated Cartoons

Martin Côté Pierre-Marc Jodoin Charles Donohue Victor Ostromoukhov

Department of Computer Science and Operations Research
Université de Montréal
{cotema, jodoinp, donohuec, ostrom}@iro.umontreal.ca

## Abstract

In cartoon drawings, hair is a strong vehicle of personality, emotions and style. In this paper, we present a powerful procedure to imitate the appearance of cartoon-like hair with a small set of parameters. Our approach is inspired by an inking technique called feathering, which consists of drawing hatches along the hair orientation while emphasizing the highlights with ink stains. Our system provides a set of tools to facilitate the positioning of the generated hair over the hand-drawn hair while keeping an intuitive interface for the artists. We can also achieve hair animation by interpolating various keyframes set by the user. Our approach has proven successful and flexible enough to adapt itself to different styles and is particularly well-suited for traditional animators.

**CR Categories:** I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture.

**Keywords:** Hair rendering, non-photorealistic rendering, animation.

## 1 Introduction

In the past few years, hair has been subject of thorough research in computer graphics due to the rendering problems it involves and the challenges that occur in its animation. The techniques developed by many authors usually focus on generating photorealistic hair, and on animating this hair using complex physical models. Some of these techniques have proven to be quite successful (see references in Section 2).

However, very few authors suggest solutions to render non-photorealistic hair such as that produced in the cartoon industry, even though the same challenging problems occur. Hair on characters can reveal many important elements about their personality and emotions. Furthermore, producing high-quality hair on such characters is becoming more problematic with the competitive and growing industry of animated features. The number of real hair strands on a head can reach huge counts. In cartoons, artists do not try to draw every strand separately, but approximate the visual effect they produce together. In this context, modeling of hair is made by putting side by side large uniform patches such as those in Fig. 1. Furthermore, a modern technique to simulate lighting is called *feathering*, which consists of drawing strokes (hatches) in the direction of the hair strands while emphasizing the highlights with ink stains (see Fig. 3). This technique helps produce convincing results, thanks to the drawing creation pipeline used in the cartoon industry.

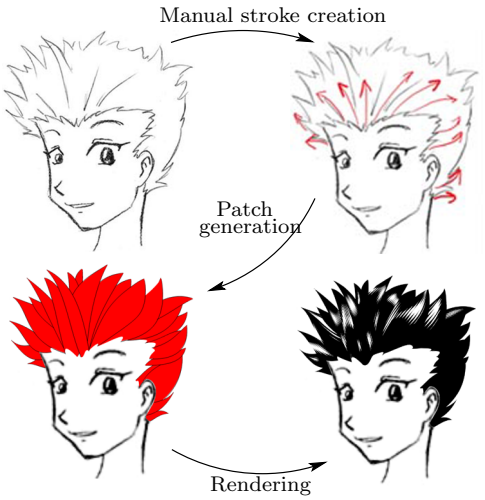



Figure 1: Operations required by our system to produce cartoonish hair from an input penciled image. Original image from Baka neko [bak n. d.].

Our contribution is to simplify the artist's work on the hair creation, following the same pipeline used in the cartoon industry. We try to automate the inking step (see Fig. 2b) in the drawing creation, while leaving flexibility to the artist. We also try to simplify the strands' representation in order to make the hair suitable for animation, thereby reducing the amount of work required by the artists to create hair movements.

To achieve this, we represent a group of hair strands with similar properties (such as length and orientation) with simple bicubic patches. These patches are generated from a pair of strokes produced by the artist, preferably following the hand-drawn hair. The final position and shape of the patch can be tweaked easily using free form deformation techniques. The inking effect is then produced with a hatching technique inspired by the work of Ostromoukhov [Ostromoukhov 1999], which enables the creation of ink stains similar to the feathering technique used by artists. This emphasizes the different highlights in the hair in an expressive manner.

The rest of the paper is organized as follows: Section 3 provides essential information about the image creation process usually used in the cartoon industry and also describes some popular techniques used by artists to produce high-quality hair. Section 4 then describes our method to automate the inking step and some modeling simplification techniques. Finally, Section 5 presents a simple solution to achieve coherent animations with our hair model, while keeping the style chosen by the artist.

## 2 Related Work

Much work has been done on photorealistic rendering of hair. Some authors attack the problem directly by modeling every hair strand individually [Rosenblum et al. 1991; Anjyo et al. 1992]. Others use hair coherence to model hair wisps [Plante et al. 2002; Koh and Huang 2001] while others use multiresolution hierarchical models [Kim and Neumann 2002; Bertails et al. 2003]. More recently, Marschner et al. [Marschner et al. 2003] developed a hair rendering system based on the micro-geometry of hair which produced highly realistic renderings. All these approaches have their pros and cons but they are usually computationally intensive, particularly for detecting collisions or computing shadows.

However, few authors have tried to render non-photorealistic hair as produced in the cartoon industry. One of the few works on the subject was done by Kowalski et al. [Kowalski et al. 1999], who produces non-photorealistic fur with procedural textures called *graftals*. The technique has a programmatic approach to hair modeling, as opposed to an artist's approach which our technique tries to follow. Furthermore, the graftal technique focuses on object silhouettes, whereas our technique also treats the shading of the hair wisps.

## 3 Cartoon Production Pipeline

The cartoon industry has evolved a lot in recent years, with the tools used by artists becoming more precise, and more effective. The drawing production process has also attained some standards in order to improve productivity. A schematic diagram of this drawing production pipeline is illustrated in Fig. 2. Ideally, the three steps of the pipeline are produced by the same artist. However, due to time constraints, this is usually not possible. The following paragraphs provide an overview of each step: *penciling*, *inking*, and *coloring*.

**Penciling.** The penciling step is where the artist draws a primary sketch of the picture and gives the rough shapes of the different items. The artist also determines what the final result will look like by specifying with rough marks where the inking stains should appear and what colors the different items should be. He therefore works closely with the other artists (the inker and the painter) to achieve the best results possible. This step is usually done with a pencil in order to make minor corrections easily. Some software solution can be used to enhance the preliminary result of the drawing process, but the main work has to be done manually. Fig. 2a shows a typical drawing that results from this step.

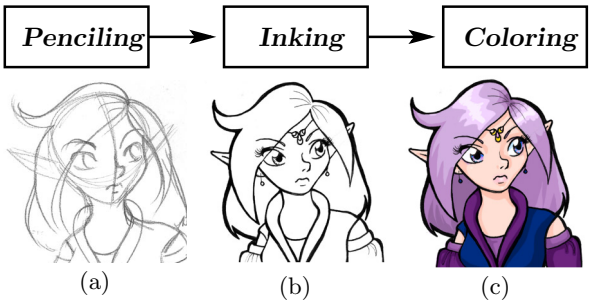



Figure 2: The drawing production pipeline usually used in the cartoon industry.

Figure 3: Two different examples of the feathering effect used to simplify and enhance the appearance of hair in cartoon drawings. (Pictures drawn by Ed Benes and Vince Russell.)

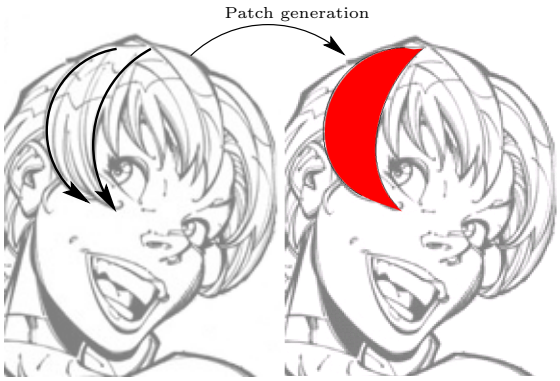



Figure 4: Patch creation from a pair of strokes. The control points of the cubic curve fitted on the sampled points of the curve are interpolated to generate a patch.

**Inking.** Inking is probably the most important step toward high-quality renderings of cartoons. Indeed, as shown on Fig. 2b, this step provides additional information about the edges that were not visible after the penciling process. The main contours are emphasized by thicker lines while details are described with thinner lines. Also, the highlights of shiny surfaces are more accentuated given the contrasts provided by the black ink. We shall see that these contrasts are essential when rendering non-photorealistic hair. The inking step is usually done by a different artist than the one who performed the penciling. This is a time-consuming process prone to error. Also, it is usually done manually since the inker has to follow the penciler's instructions about where to put the ink stains and which width to use for the different lines.

**Coloring.** Unlike the previous steps, the coloring step is often done with software solutions. Choosing the right colors for a given picture is an art of its own, and the duty of the artist. Using software solutions for this step can provide much more interesting results since the artist is able to test some color configuration before actually painting the final drawing. Fig. 2c shows a software coloration of an inked picture.

A drawing pipeline such as the one illustrated in Fig. 2 helps many artists work in parallel on a specific animation, but does not simplify the creation of complex drawings. This is particularly true when trying to model expressive hair for the characters. The tiny geometry in hair strands makes their drawing difficult. Furthermore, the creation of con-

vincing hair is essential to convey the proper personality and emotions of the different characters. For this reason, artists have developed different techniques to simplify the drawing of hair, while keeping its overall aspect.

One of the most popular techniques used by artists is called *feathering*. This technique is useful for enhancing highlights in a scene, but is particularly suitable for hair. The principle is quite simple: the artist emphasizes the darker regions of the material with ink stains, which indirectly emphasize the highlights. The different tones are achieved by drawing hatches (or cross-hatches) of variable width. This is a powerful way to simplify hair representation, as the hatching shows the general direction of the strands while giving a good approximation of the anisotropic properties of hair. Examples of the feathering technique are illustrated in Fig. 3. Note that the inking in these examples is sufficient to describe many properties of the characters' hair.

# 4 Feathering System

## 4.1 Overview

This section presents our software solution to model hair wisp and reproduce the feathering effect. First, however, we need to briefly describe the different steps required to produce cartoonish hair. They are illustrated in the following diagram and discussed more thoroughly later.

A preliminary picture drawn by the artist is required in order to fit our digitized hair.

Some patches are positioned over the hand-drawn hair in 2D (see Section 4.2).

Some *pseudo-lights* are then positioned to get the proper highlights. At this stage, different parameters can be modified in order to get the desired visual effect (see Section 4.3).

Finally, the planar look of the positioned patches can be broken using a depth perturbation technique (see Section 4.5).

## 4.2 Interactive Modeling

Reproducing hand-made inking effect created for cartoons is an indispensable step toward achieving high-quality rendered hair. However, it is crucial to provide good modeling tools to help the users during the inking creation. Even if modeling will always be done by artists, it is desirable to simplify their work.

A hair wisp [Plante et al. 2002; Koh and Huang 2001] is defined as a group of hair strands with similar properties such as color, shape and orientation. The creation of a wisp is done simply by defining a pair of silhouette curves from which the patch is created (see Fig. 4). The strokes are usually drawn over the penciled image and follow the hair wisps. Each stroke is then converted to a cubic curve by fitting it on the stroke's sampled points. The patch is then created by addition of the missing control points using a linear interpolation between the two curves.[1] The result of this process might not be perfect, but certainly provides a good starting approximation.

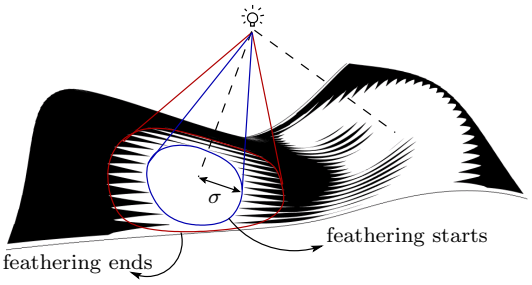



Figure 5: A 3D patch with a pseudo-light source that produces the different highlights. The offset is used to vary the highlight sizes.

The artist can tweak the generated patch to get a better fit with the penciled image, or to provide some artistic effect. To this end, the artist can manually change the position of any control point. However this approach is time consuming and not very intuitive. A better solution here proposed is to use free-form deformation techniques (FFD), introduced by Barr [Barr 1984], to manipulate the patch position either globally or locally. In this perspective, the system allows the user to select a region of interest he wants to manipulate. A bicubic patch is then created around the selected area and the modification of the FFD patch indirectly modifies the selected control points' position in a simple and intuitive way.

## 4.3 Rendering

In our technique, the shading of the hair is inspired by the work of Ostromoukhov [Ostromoukhov 1999], who reproduces the engraving done manually by artists. His solution requires fitting several bicubic patches over an input image. The grooves on the patches are then rendered by thresholding the intensity of the corresponding image sub-regions. This approach is powerful, but unsuitable in our case since our input image is a penciled drawing without enough intensity variations to derive the proper hatching width. Furthermore, it would be desirable to give the artist the ability of controlling the highlights position without regard to an input image.

In our technique, the shading is done in the *feathering* style. This requires a source of "light" to modulate the hatches' width. These pseudo-lights are positioned by the system in 3D around the patches according to the artist's choice in 2D. Fig. 5 shows a pseudo-light with the corresponding feathering effect.

The actual feathering effect is performed by using a series of polygons uniformly distributed over the patch and whose size depends on the pseudo-lights' position. Technically, since every hair wisp is modeled with a bicubic patch made of two axes (namely $u$ and $v$), a set of sampling points $p_{u,v}$ are uniformly distributed over the wisp. From this distribution, a polygon is created at each sampled point $p_{u,v}$ based on its three neighbors $p_{u,v+1}$, $p_{u-1,v}$ and $p_{u-1,v+1}$. As shown in Fig. 7, the width of each polygons depends on the magnitude of vector $\vec{m}_{u,v+1}$ and $\vec{m}_{uv}$. The magnitude of

[1] The patch is actually created from a *skin* operation provided by a nurbs surface library called *NURBS++* [Lavoie n. d.].

these two vectors depends on the $\vec{n}_{u,v} \cdot \vec{l}$ factor where $\vec{n}_{u,v}$ is the normal at $p_{u,v}$ and $\vec{l}$ is the pseudo-light position. In this way, a polygon directly illuminated by a pseudo-light has a small width whereas an indirectly illuminated polygon has a larger width.

Mathematically, the vector $\vec{m}_{uv}$ is calculated as follows:

$$\vec{m}_{u,v} = (p_{u-1,v} - p_{u,v}) \max\left(0, 1 - (\vec{n}_{u,v} \cdot \vec{l}) - \sigma\right) \ \omega. \quad (1)$$

where $\sigma$ and $\omega$ are parameters for fine tuning the shading effect (see Fig 5 and 8). More specifically, $\omega$ makes the feathering grow more or less quickly whereas the offset $\sigma$ allows to specify the highlights diameter. Let us notice that vector $\vec{m}_{u,v+1}$ is also computed with Eq. 1.

The polygon located over the sampling point $p_{uv}$ is thus rendered with the following coordinates: $p_{u,v+1}, p_{u,v}, (p_{u,v} + \vec{m}_{u,v})$ and $(p_{u,v+1} + \vec{m}_{u,v+1})$. As shown in Fig. 7, once the latter polygon is rendered, a symmetrical polygon is created at $p_{u,v} - \vec{m}_{u,v}$. These two polygons are refered to as the *feathering polygons*.

An interesting aspect about this technique is that the tone of the patch is independent of the frequency of hatching strokes on it (i.e., the number of strokes per linear units). Raising the sampling resolution in the $u$ direction decreases the width variations of the feathering since the neighboring strokes are nearer. The actual implementation is interactive and was made with OpenGL[Woo et al. 1999].

### 4.4 Pseudo-lights Positioning

It is important to provide a simple interface to create highlight spots, since many of them may be needed for a single wisp. Direct light positioning in 3D space would prove too complex for this task. Our system provides a simple user interface, in which the user simply places a circular region over the wisp where he wants the highlight, then the system automatically produces a properly positioned pseudo-light. This is done by translating the circle's center along the surface's normal. The light's distance is proportional to the circle's radius.

### 4.5 Planar Aspect Removal

Since the strokes are drawn in screen space, the different patches may have a planar look, which might not adequately represent the subtlety of hair. To solve this, the proposed software allows to perturb the $z$ coordinates of the control points and thus break the planar aspect of the wisps. To spare the user the trouble of independently moving the control points in 3D (which would be counterintuitive in the context of a 2D drawing system), it was decided to implement a simple *click-and-hold* interface. In this way, whenever the user wants to change the depth value of a given point, he simply has to click and hold the point. The longer the user holds the point, the farther (or nearer) it gets. This simple interface turned out to be remarkably efficient since it allows instant feedback of the $z$-perturbations without having to change the current point of view. The same technique is used to change the light depths at interactive framerates.

## 5 Animation

Our representation of hair can easily be extended to animation. Different solutions are conceivable to create realistic movements of wisps, but we opted for a simple keyframe interpolation (or *in-betweening*) technique. This approach is widely used in the cartoon industry to animate characters of any kind. The way this in-betweening animation technique works is simple. First, the penciler creates pictures over some specific keyframes. Once these keyframes are drawn, other artists (called *in-betweeners*) interpolate it manually. This technique is known to be time consuming and error prone. For this reason, our system allows animation to be done automatically. Our software simply adjusts the hair wisps over every keyframe provided by the penciler and then automatically interpolates the control points of the patches between each keyframe. The accompanying video shows an example of animation with keyframe interpolation.

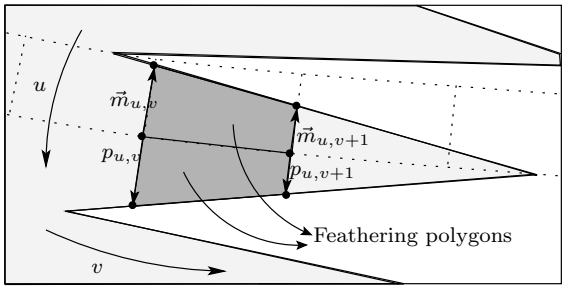



Figure 7: Process of the feathering creation. At each point $p_{u,v}$ on the patch, a new polygon is created. The feathering polygon is created by computing the slope $\vec{m}_{u,v}$ at $p_{u,v}$.

## 6 Results

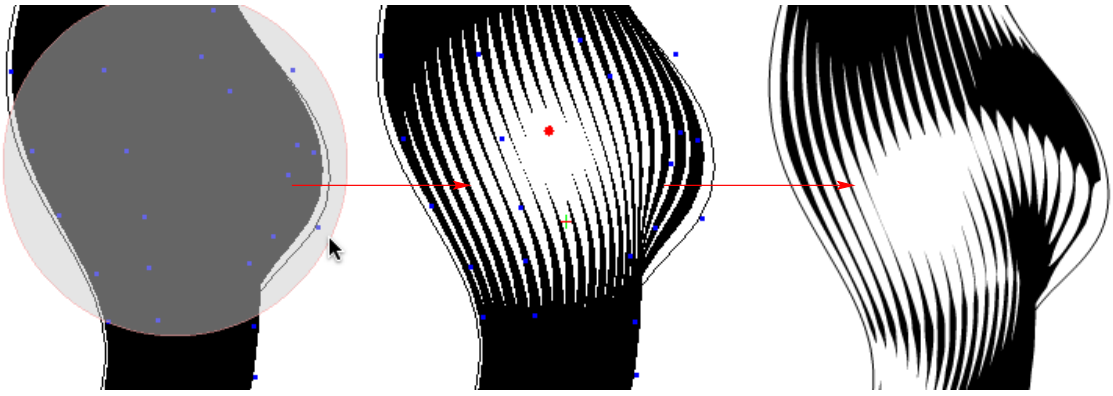
Our inking system provides expressive visual effects while giving flexibility to the user, as shown on Fig. 8. Furthermore, because of the intuitive modeling interface and the fact that the overall number of parameters to manipulate is small, our system allows to easily and quickly create different hair styles. For example, if one wants smooth highlights without anisotropic effects, a large value for the $\omega$ parameter would make the feathering grow more quickly and reduce the hatching effect. Similarly, the user can underline tone variations by making the hatches grow more slowly with the $\sigma$ parameter.

The hair wisps are created using the patch approximation tool provided by a nurbs library [Lavoie n. d.]. The patches can then be tweaked either by direct control point manipulation or by global editing with FFD patches. The interactive manipulations of the patches with direct visual feedback is clearly observed. Additional results are illustrated in Fig. 9 while the accompanying video shows our interface in action. To obtained results shown in Fig. 11, it took an untrained artist less than 10 minutes.

## 7 Discussion and Future Work

In this contribution, a framework to model and render non-photorealistic cartoonish hair has been presented. Our system provides results comparable to the inking made by cartoon artists, and thus could enhance the work flow in the industry with regards to inking. Our polygon-based solution to recreate feathering allows the users to interactively build and edit many strands and still get interactive feedback from the interface.

Modeling hair wisps with pairs of strokes has shown to be powerful yet easy to use. With this interface, the user can create good preliminary patches in a very short period of



(a) (b) (c)

Figure 6: Technique described in Section 4.4 to generate highlight spots over a wisp. In (a), a circular region is manually placed by the user. A the pseudo-light is then automatically positioned (in (b)) such that its shading fits the overall the region given in (a). In (c), the user has performed some depth perturbations (Section 4.5) to give a 3D look to the highlights.


Figure 8: Results of the modification of the $\omega$ (left to right) and $\sigma$ (top to bottom) parameters. Incrementing the slope ($\omega$) provides bigger ink stains while incrementing the offset ($\sigma$) generates larger highlight spots. These parameters are sufficient to describe many different inking styles

time. The FFD manipulation tool is also a useful instrument to help the user fine tune the wisps' shape.

In the actual implementation, the hair wisps and the pseudo-lights can be animated with a keyframe technique. In this perspective, the user fits the patches (and the pseudo-lights) on some hand-drawn keyframes made by another artist. The animation is made by simply interpolating the control points between the keyframes. The coherence in animation is easily achievable either by using a predefined set of lights or by moving the lights according to hair movements.

Aliasing is still a problem with our system. Since we use OpenGL to render the hair, there is no way to completely remove the aliasing caused by the small feathering polygons. However, the effect of aliasing can be significantly reduced by supersampling the scene. We plan to use a vector-based output format to render the final results, while keeping OpenGL to achieve interactive feedback for wisp creation and editing. This vector output format could be used in standard vector drawing packages for final touch-ups.

Our long-term goal is to extend our system to a more generic rendering system that would allow the creation of inking for an arbitrary scene. The same feathering technique is sometimes used by artists to create penumbra regions caused by shadows (see Fig. 10). Although there is a lot of progress to be done until we achieve this, it is our believe that the current framework brings new light on the everlasting problem of cartoon style synthesis.

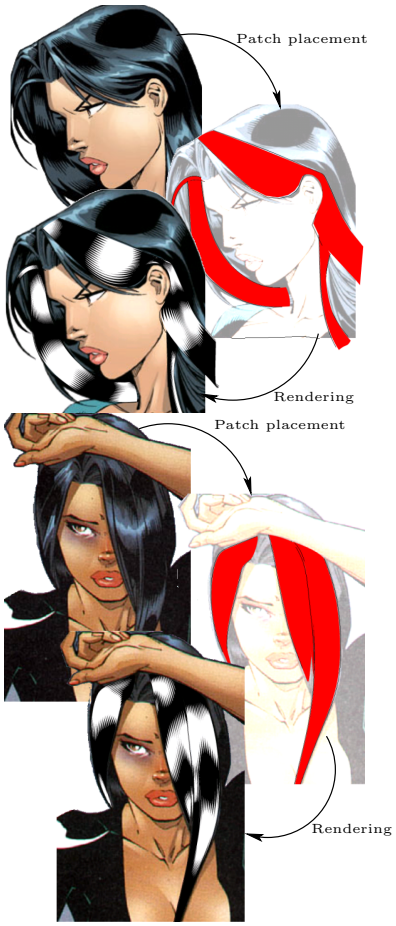



Figure 9: Results of applying some patches over the hair and by rendering using our inking system.

Figure 10: Example where the feathering technique is used to create varying tones for shadows. We could possibly extend our system to produce this kind of effect. Source of the image: [Alanguilan n. d.].

## References


Alanguilan, G. Art tips. http://www.laguna.net/∼timawa/tips2.htm.

Anjyo, K., Usami, Y., and T.Kurihara. 1992. A simple method for extracting the natural beauty of hair. In *Proceedings of SIGGRAPH 92*, 26(2):111–120.

Baka neko: Anime drawing tutorials. http://www.cat-print.com/howto/.

Barr, A. H. 1984. Global and local deformations of solid primitives. In *Proceedings of SIGGRAPH '84*, 21–30.

Bertails, F., Kim, T.-Y., Cani, M.-P., and Neumann, U. 2003. Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. *Symposium on Computer Animation '03* (July).

Kim, T.-Y., and Neumann, U. 2002. Interactive multiresolution hair modeling and editing. *Proceedings of SIGGRAPH '02*, 620–629.

Koh, C. K., and Huang, Z. 2001. A simple physics model to animate human hair modeled in 2d strips in real time. *Computer Animation and Simulation '01*.

Kowalski, M. A., Markosian, L., Northrup, J. D., Bourdev, L., Barzel, R., Holden, L. S., and Hughes, J. 1999. Art-based rendering of fur, grass, and trees. *Proceedings of SIGGRAPH 99*, 433–438.

Lavoie, P. Nurbs++ library. http://libnurbs.sourceforge.net.

Marschner, S. R., Jensen, H. W., Cammarano, M., Worley, S., and Hanrahan, P. 2003. Light scattering from human hair fibers. *Proceedings of SIGGRAPH 03*.

Ostromoukhov, V. 1999. Digital facial engraving. In *Proceedings of SIGGRAPH 99*, 417–424.

Plante, E., Cani, M.-P., and Poulin, P. 2002. Capturing the complexity of hair motion. In *Graphical Models*, vol. 64, 40–58.

Rosenblum, R., Carlson, W., and Tripp, E. 1991. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *The Journal of Visualization and Computer Animation* (October – December), 2(4):141–148.

Taeha. Inking and cleaning up penciled drawings. http://www.artlair.com/startart/tut_inking.html.

Woo, M., Neider, J., Davis, T., and Shreiner, D. 1999. *OpenGL Programming Guide*. Addison Wesley.

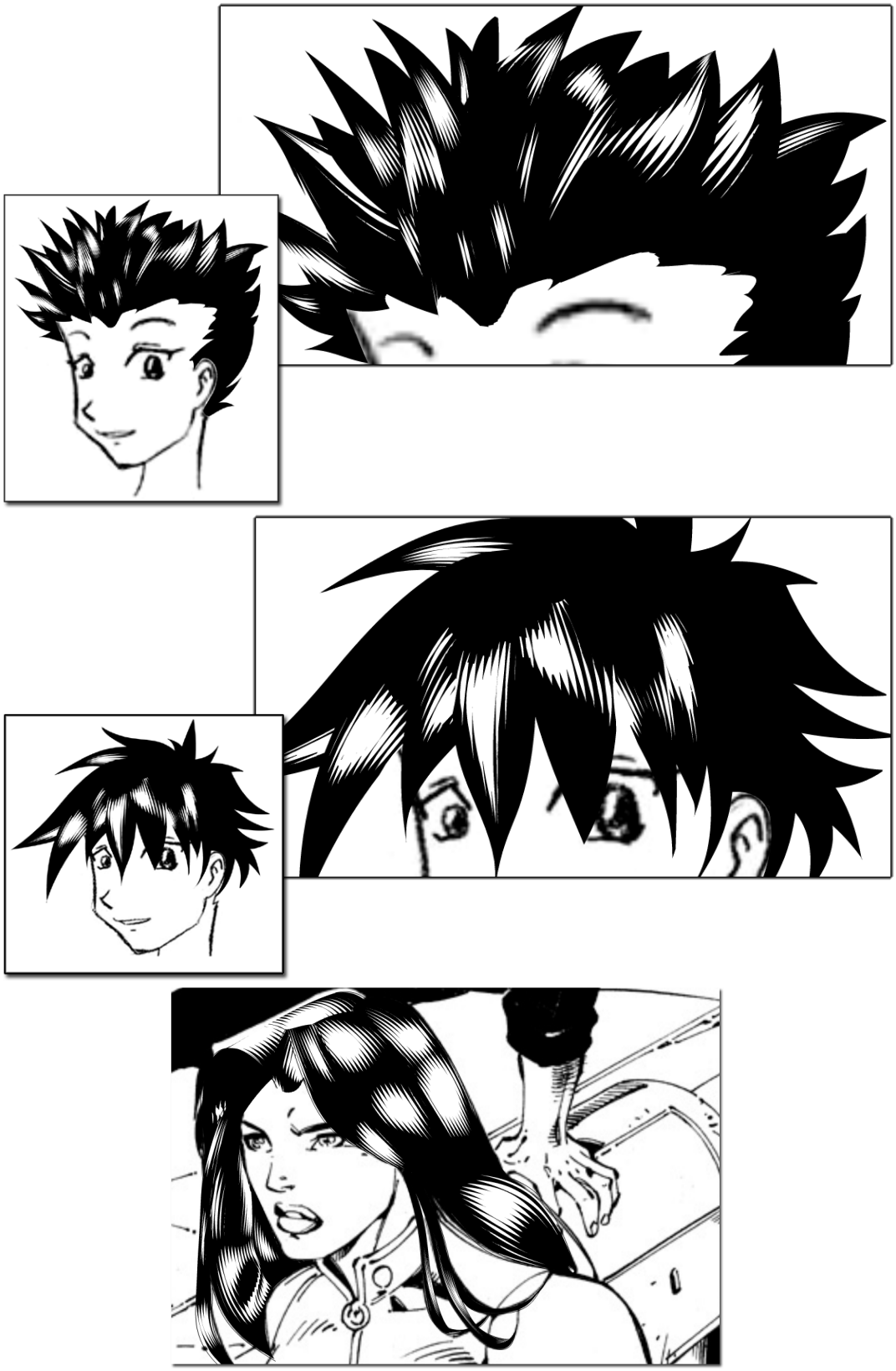


Figure 11: Results obtained with our technique.