濡れた毛のリアルタイムレンダリング
Real-Time Rendering of Wet Fur

パウロ シルバ† 坂東 洋介†‡ 陳 炳宇§ 西田 友是‡
Paulo SILVA†, Yosuke BANDO†‡, Bing-Yu CHEN§ and Tomoyuki NISHITA†

† 東京大学 † The University of Tokyo
‡ 東芝 ‡ Toshiba Corporation
§ 国立台湾大学 § National Taiwan University

E-mail: †{paulo,ybando,nis}@is.s.u-tokyo.ac.jp, §robin@ntu.edu.tw

1 Introduction

Fur is present in most mammals which are common characters in both movies and video-games, and it is important to model and render fur both realistically and fast. In computer graphics, there are mainly two approaches to fur rendering. Using primitives such as lines or curves, and using textures containing sampled fur geometry. The former case is oriented towards realism and the latter case focusing on real-time rendering performance.

However, the existing research focused on real-time rendering leaves out important factors such as the clumping effects visible when fur becomes wet or dries up.

In this paper we describe a method to simulate effects visible in water-fur interaction such as the change in color and shape of the fur (see Fig. 1 and Fig. 2).

2 Related Work

Fur rendering using textures was first presented by Kajiya and Kay [1]. Lengyel et al. [2] presented a real-time method using texture layers based on the work by Neyret [3]. Recently several authors improved this method in both performance [4, 5] and rendering effects like shadows [6] or weathering fur [7]. A method for rendering wet fur clumps was proposed by Bruderlin [8]. This is a primitive based method which concentrates primarily on the shape the fur takes when wet.

However, none of the existing methods produces wet-like fur effects such as clumping in real-time.

3 Wet Fur Effects

In our method, the fur representation is based on Lengyel et al.'s [2] work. Our fur clumping method conceptually resembles Bruderlin's [8] work, but is oriented towards a GPU implementation for real-time rendering. We adapt the work by Goldman [9] to compute the fur color while using a texture based representation for the fur.

Our method uses as input a mesh and a texture representing the fur color. We create a wetness mask texture (Fig. 3), which contains the clump regions information. Here we assume a parameterization from the model to the texture is available. In the wetness mask we store the center of the clump $\vec{c}$, its radius r, and its wetness or clump-percent ρ (see Sec. 3.1).

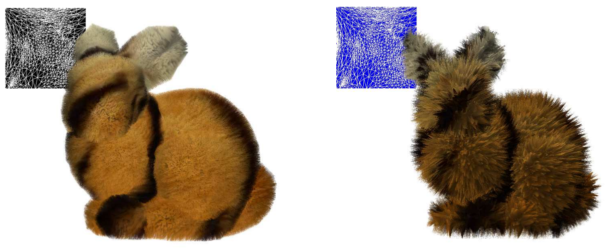
(a) (b)

Figure 1: Example of real-time fur visuals possible with our method. (a) an initially dry Stanford Bunny, (b) and the same model soaking wet. Notice the differences in fur clumping shape and color. The top-left square represents the wetness texture used by our method to control the fur shape.

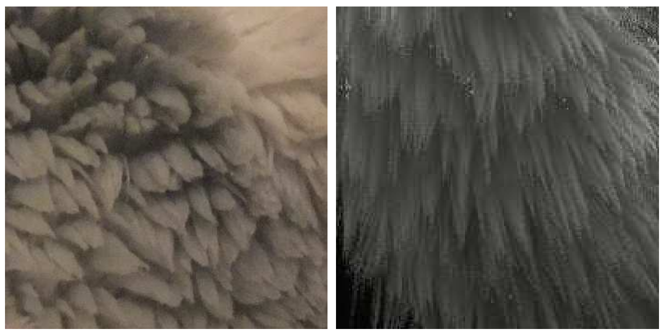
(a) (b)

Figure 2: Comparison between: (a) real clumping of a wet blanket and (b) the result produced by our method.

During rendering, in the fragment shader we apply a fur clumping process as described in Sec. 3.2 followed by the color computation as described in Sec. 4.

3.1 Clump Mask Generation

The wetness mask (Fig. 3 (a)) is used to specify wet regions. For a clump located at surface parameter $\vec{c} = (u, v)$ with radius r, and a clump-percent $\rho \in [0, 1]$, the center position on the model is computed from the parameterization, and written to the wetness mask in the texture channels (R, G). Additionally we write (r, ρ) to texture channels (B, A).

A droplet is blended into the wetness mask texture using Algorithm 1.

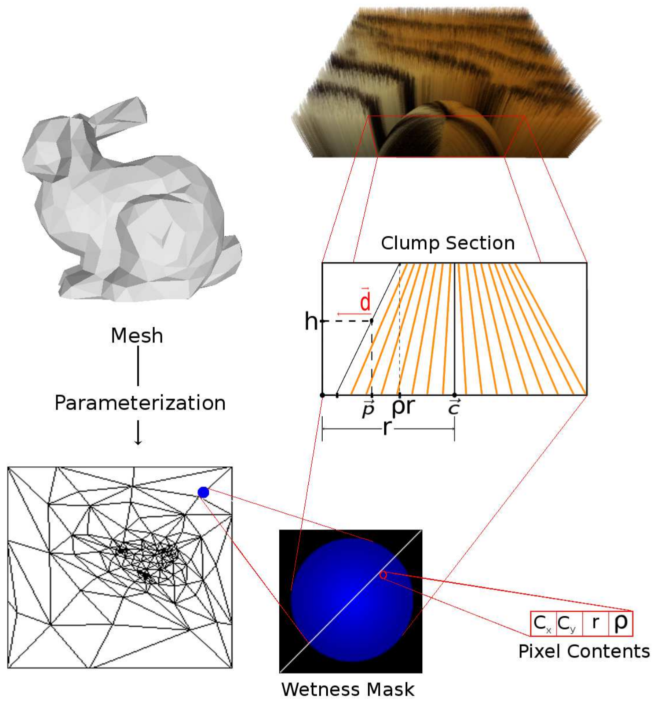

Figure 3: Using the mesh parameterization and a wetness mask, we compute the displacement $\vec{d}$. The black area in the wetness mask represents dry fur. The colored area represents the wet area. In the clump section view we can see: $\vec{d}$ is the displacement vector, $\vec{p}$ is the current position in the wetness mask, $\vec{c}$ is the center of the clump, h is the height of the current texture layer, r is the radius of the clump, ρ is the clumping percentage, ρr is the distance from the center of the clump $\vec{c}$ outwards.

3.2 Fur Clumps Deformation

To deform the fur in order to create clumps, we compute a displacement $\vec{d}$ in the fragment shader for each visible position on the input mesh. Then we displace the texture coordinates of that position if the region is wet. To check whether the region is wet or not, we inspect the wetness mask (Fig. 3) using the model parametrization. The displacement $\vec{d}$ is illustrated schematically in the clump cross-section view in Fig. 3. There are many possibilities for displacement function as the one presented in [8]. Here we compute the displacement $\vec{d}$ using Eq. 1 bacause we believe this produces an interesting result, but any other function would also be possible.

$$\vec{d}(h) = ((1-w)h + w(1 + k^{-1}\ln h))\rho r(\vec{p} - \vec{c}), \qquad (1)$$

where $h \in [\Delta h, 1]$ is the normalized fur length, $\Delta h = 1/N$ where N is the number of layers, ρ is the wetness, r is the clump radius, $\vec{p}$ is the current pixel position on the wetness mask, and $\vec{c}$ is the center of the clump. Notice that h does not take the value 0 because that value is considered to be the base surface which is rendered directly bypassing the algorithm. The factor w is a constant that controls how the fur shape bends towards the clump center. Finally $k = -\ln \Delta h$ is a normalization of the $\ln h$.

Algorithm 1 Blend Droplet Into Wetness Mask

```
if $\|\vec{c}_{\text{mask}} - \vec{c}_{\text{drop}}\| \leq r_{\text{drop}}$ then
  if $\rho_{\text{mask}} \neq 0$ then
    $\rho_{\text{total}} = \text{clamp}(\rho_{\text{mask}} + \rho_{\text{drop}}, 0, 1)$
    if $\rho_{\text{total}} \neq 0$ then
      $\alpha = \rho_{\text{drop}} / \rho_{\text{total}}$
      $\vec{c}_{\text{mask}} = (1-\alpha)\vec{c}_{\text{mask}} + \alpha\vec{c}_{\text{drop}}$
      $r_{\text{mask}} = (1-\alpha) r_{\text{mask}} + \alpha r_{\text{drop}}$
    end if
    $\rho_{\text{mask}} = \rho_{\text{total}}$
  end if
else
  $\vec{c}_{\text{mask}} = \vec{c}_{\text{drop}}$
  $\rho_{\text{mask}} = \rho_{\text{drop}}$
  $r_{\text{mask}} = r_{\text{drop}}$
end if
```

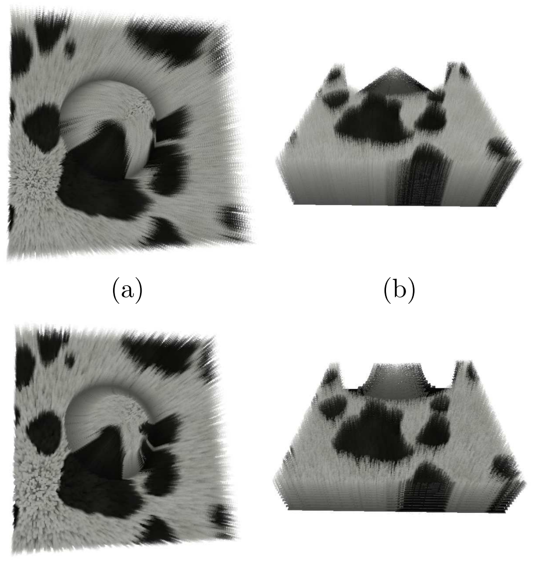

Figure 4: Fur with $w = 0$ (a,b) and $w = 1$ (c,d).

Since h takes values between Δh and 1, the $|\ln \Delta h|$ is the largest value that function takes, and therefore is used in the normalization. Since the values of $k^{-1}\ln h$ are $\in [-1, 0]$ we add 1 to move the interval to $[0, 1]$. This blending between linear and logarithmic factors produces the shapes illustrated in Fig. 4 (a,b) and (c,d). In Fig. 4 (a,b) the fur maintains a straight shape while leaning towards the clump center, while in Fig. 4 (c,d) the fur shape bends in towards the center of the clump along its length. If an area is not in a clump region, we do not displace any texture coordinates, and directly render the fur using the original texture coordinates. This idea is stated in Algorithm 2, where $\vec{uv}_{\text{color}}$ and $\vec{uv}_{\text{geo}}$ are the corresponding color and fur texture coordinates. Note that although the fur texture has three coordinates (u, v, h), only two (u, v) are displaced.

Applying the wetness mask in Fig. 5 (b) to Fig. 5 (a), the desired clumping effect is achieved as shown in Fig. 5 (c).

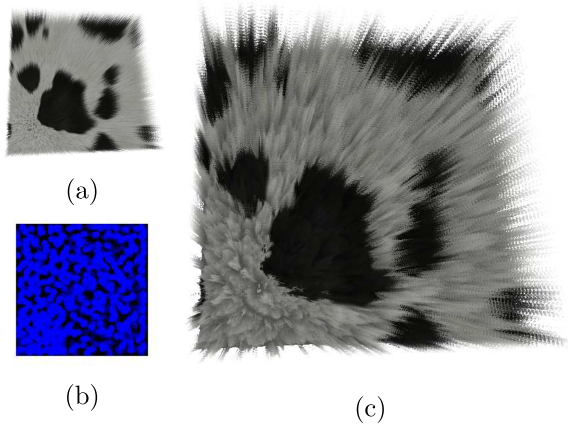

Figure 5: (a) Input fur texture. (b) Wetness mask with wet areas displayed in blue. (c) The result of applying to input fur texture (a) the method presented in Sec. 3.2, using the wetness texture (b).

4 Rendering

In the fragment shader, we select the corresponding texture fragment from the fur texture layers (3D texture) using the interpolated surface texture coordinates (u, v) displaced by $\vec{d}(h_i)$ (see Eq. 1) and third texture coordinate h_i, which is the current normalized fur length $h_i = (1 + n_i)/N$, where $n_i \in [0, N-1]$ is the current layer. For non-transparent fragments, we use an illumination model adapted from [9]. For a given specular color K_s and with the color K_{color} extracted from the input color texture, we compute the final fragment color C_{frag} as:

$$\begin{aligned} C_d &= K_{\text{color}} \max(0, \vec{l} \cdot \vec{n}), \\ C_s &= K_s I_s, \\ C_{\text{frag}} &= \gamma_{occ}\gamma_{wet}(C_d + C_s). \end{aligned} \tag{2}$$

where $\vec{l}$, $\vec{n}$ are the light direction and normal vector. Here we use the surface normal and not the fur normal since the objective is to shade the surface accordingly to its shape, but the fur normal is used to compute the specular contribution indirectly through the use of the fur tangent. For the specular contribution I_s we use the method by [9] with added water contribution for attenuation as stated in Eq. 3.

$$\begin{aligned} Is(\vec{t}_{ws}, \vec{l}, \vec{e}, \rho, p) &= [(\vec{t}_{ws} \cdot \vec{l})(\vec{t}_{ws} \cdot \vec{e}) + \\ &\quad \sin(\vec{t}_{ws}, \vec{l}) \sin(\vec{t}_{ws}, \vec{e})]^{k_{spec}(\rho)p}, \\ k_{spec}(\rho) &= 1 + \sigma_{spec}(\rho - 1). \end{aligned} \tag{3}$$

Where $\vec{t}_{ws}$ is the fur tangent in world space coordinates as given by Eq. 4, $\vec{l}$ is the light direction, $\vec{e}$ is the eye direction, ρ is the water percentage as previously stated, and p is the specular exponent. The constant σ_{spec} is user given and controls how much the wetness ρ contributes to the specular exponent.

$$\begin{aligned} \vec{\Delta d}(h_i) &= \vec{d}(h_i) - \vec{d}(h_{i-1}), \\ \Delta h &= 1/N, \\ \vec{t}_{ts} &= (\Delta d_x, \Delta d_y, \Delta h), \\ Q_{ws} &= [\vec{t}\, \vec{b}\, \vec{n}], \\ \vec{t}_{ws} &= Q_{ws}\vec{t}_{ts}/\|\vec{t}_{ts}\|. \end{aligned} \tag{4}$$

The term $\vec{t}_{ts}$ represents the vector tangent to the fur strand in texture space, Q_{ws} is the base change matrix, and $\vec{t}$, $\vec{b}$ form the surface tangent space while $\vec{n}$ is the surface normal. We add a simple occlusion and darkening due to the wetness by introducing into Eq. 2 the factors:

$$\begin{aligned} \gamma_{occ} &= 1 + \sigma_{occ}(h_i - 1), \\ \gamma_{wet} &= 1 - \sigma_{wet}\rho. \end{aligned} \tag{5}$$

The value $\sigma_{occ} \in [0, 1]$ and $\sigma_{wet} \in [0, 1]$ are constants given by the user, which control the percentage of hair darkening due to occlusion and wetness respectively. Although methods such as [10, 11, 12] can possibly provide better results, for short fur this simple method is effective and inexpensive.

Algorithm 2 Fur Clumping

```
if $\rho_{\text{mask}} \neq 0$ then
    compute $\vec{d}$
    if $\|\vec{c}_{\text{pixel}} - \vec{c}_{\text{mask}}\| > r_{\text{mask}}$ then
        ignore pixel from a different clump
    end if
    if $\rho_{\text{mask}} = 0$ at $\vec{c}_{\text{pixel}} + \vec{d}$ then
        ignore pixel from a dry area
    end if
    $\vec{uv}_{\text{color}} \leftarrow \vec{uv}_{\text{color}} + \vec{d}$
    $\vec{uv}_{\text{geo}} \leftarrow \vec{uv}_{\text{geo}} + \vec{d}$
    if no geometry at $\vec{uv}_{\text{geo}}$ then
        ignore transparent pixel
    end if
    compute color using $\vec{uv}_{\text{color}}$
end if
```

5 Results

We conducted performance tests on a *Intel Quad Core^{TM} 3GHz*, with 2*GB* of RAM, and with a *GeForce 8800 GTS* with 320*MB* video memory. The resolution of the wetness mask and fur texture layers is 256 × 256. The frame-rates obtained are summarized in Table. 1. Our results indicate that as the model size (number of triangles) increases, the resolution of the screen influences less the frame-rate. However we still obtain real-time performance for most of the cases. In Fig. 6 we can see some examples of our method applied to some input models.

6 Conclusion and Future Work

We presented a method to add wet-like fur effects to texture based fur representations. Our method introduces

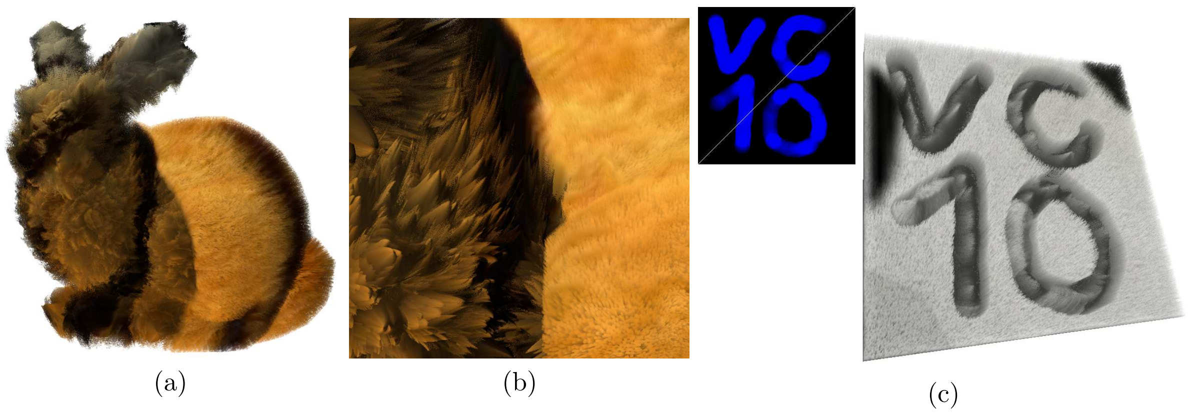

Figure 6: (a) a Stanford bunny partially covered with wet clumped fur and partially with dry fur. (b) a detailed view of the area between the wet and dry fur. Notice the differences in color, and shape. (c) the fur clumps form the letters "VCS 10".

Table 1: Performance values by our method in our tests. Even for large meshes we obtain real time performance. The "layers" field indicates how many times the model is redrawn to produce the fur effect as explained in [2].

16 Layers			
Model	#Triangles	800 × 600	1680 × 1050
Hippo	44.452	154	80
Cow	92.689	78	63
Bunny	142.307	24	22
64 Layers			
Hippo	44.452	51	24
Cow	92.689	28	21
Bunny	142.307	8	8

a texture to represent fur clumping information, such as clump position, radius, and amount of water stored at a particular point on an input surface. With this we can render interactively effects such as hair clumping due to water accumulation.

When fur becomes overly wet the clumps disappear, and the fur forms a smooth surface with high specular reflection. In our current method we do not support this transition, which is therefore left for future work. We think that an interesting extension would be to add support to high detail contact interactions with the fur texture. That is, to support real-time deformation of fur as a result of a contact with complex shapes, such as hands, etc.

References

[1] J. T. Kajiya and T. L. Kay, "Rendering fur with three dimensional textures," in *ACM SIGGRAPH 1989 Conference Proceedings*, pp. 271–280, 1989.

[2] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe, "Real-time fur over arbitrary surfaces," in *Proceedings of the 2001 Simposium of Interactive 3D Graphics*, pp. 227–232, 2001.

[3] F. Neyret, "Modeling and animating, and rendering complex scenes using volumetric textures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 55–70, 1998.

[4] J. Isidoro and J. L. Mitchell, "User customizable real-time fur," in *ACM SIGGRAPH Conference Abstracts and Applications*, p. 273, 2002.

[5] S. Tariq and L. Bavoil, "Real time hair simulation and rendering on the gpu," in *ACM SIGGRAPH 2008 Talks*, p. Article No. 37, 2008.

[6] B. Sheng, H. Sun, G. Yang, and E. Wu, "Furstyling on angle-split shell textures," *Computer Animation and Virtual Worlds*, vol. 20, no. 2-3, pp. 205–213, 2009.

[7] S. Jiao and E. Wu, "Simulation of weathering fur," in *Proceedings of the 2009 International Conference on Virtual Reality Continuum and its Applications in Industry*, pp. 35–40, 2009.

[8] A. Bruderlin, "A method to generate wet and broken-up animal fur," *The Journal of Visualization and Computer Animation*, vol. 11, no. 5, pp. 249–259, 2000.

[9] D. B. Goldman, "Fake fur rendering," in *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 127–134, 1997.

[10] T. Lokovic and E. Veach, "Deep shadow maps," in *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 385–392, 2000.

[11] G. Yang, H. Sun, E. Wu, and L. Wang, "Interactive fur shaping and rendering using nonuniform-layered textures," *IEEE Computer Graphics and Applications*, vol. 28, no. 4, pp. 85–93, 2008.

[12] R. Gupta and N. Magnenat-Thalmann, "Interactive rendering of optical effects in wet hair," in *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, pp. 133–140, 2007.