COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1463

SPECIAL ISSUE PAPER

Adaptive surface splatting for facial rendering

Hyeon-Joong Kim[1], A. Cengiz Öztireli[2], Markus Gross[2] and Soo-Mi Choi[1]*

[1] Department of Computer Science and Engineering, Sejong University, Seoul, Korea
[2] Computer Graphics Laboratory, ETH Zürich, Zürich, Switzerland

ABSTRACT

Recent advances in facial scanning technology provide highly detailed faces, including fine wrinkles. However, because of the increasing complexity of the resulting models, it is necessary to reduce the amount of data while preserving small-scale facial features. In this paper, we propose a new adaptive surface splatting method to reduce the number of splats by optimizing the size and shape of splats using geometric and color information. Using the optimized splats, we can improve the rendering quality, especially in the visually sensitive feature areas. Our adaptive surface splatting is very effective to render massive facial data on tablet PCs or smartphones. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS
point-based rendering; surface splatting; facial rendering; spectral sampling

***Correspondence**
Soo-Mi Choi, Department of Computer Science and Engineering, Sejong University, 98 Gunja-Dong, Gwangjin-Gu, Seoul 143-747, Korea.
E-mail: smchoi@sejong.ac.kr

1. INTRODUCTION

People tend to identify one another by their facial features and perceive their emotions from facial expressions. The human optical system responds more sensitively to small facial changes than to changes in the shape of other objects. This makes the modeling of a face a challenging issue in computer graphics. In order to create a digital face that is nearly identical to a human face, we must first express facial geometry accurately. Second, there must be no errors in regions where abrupt color changes signify visually important features such as the lip, eye, and eyebrow, because viewers generally pay more attention to these regions when reading facial expressions [1].

There are now many ways to acquire facial shapes, but it is generally easier and faster than it was to obtain detailed facial data. Recently, it has become possible to capture faces instantly, including fine features such as wrinkles and pores, by combining images taken by multiple cameras [2]. As detailed shapes, textures, and measured appearance data [3,4] become available to create more realistic digital faces, the amount of data to be processed increases, and the acquisition of more data is also likely to include unnecessary or redundant information that will not influence the final images. Therefore, we need a method to simplify facial data and then render it while preserving facial features. This would allow realistic faces to be rendered on tablet PCs or smartphones, as well as improve rendering speeds on PCs.

In this paper, we propose a new surface splatting method for adaptive simplification and high-quality rendering of point-sampled surfaces. Our main contributions are as follows: first, our approach directly works on point-sampled surfaces without any connectivity information between points. In contrast to previous point-based approaches, we are not focusing only on geometric information but also consider color and facial feature information for rendering purposes. Second, we use a splat-based representation. Splat shapes are computed on the basis of the same measures used for the sampling stage, and thus they are well adapted for the preservation of important features using the resampled data set. Third, we detect facial features such as the lips, eyes, and eyebrows. This visually important feature information, as well as geometric and color information, can be used for determining the size and shape of splats, influencing the rendering results.

2. RELATED WORK

2.1. Point-based Rendering

Point-based rendering has received much attention as an alternative to polygon-based rendering because of its simplicity and flexibility. The main advantage of point-based rendering is that it does not need to take topological consistency conditions into account because no connectivity information between points has to be maintained.

Like most discretization approaches, point-based rendering involves an aliasing problem on the surface that is reconstructed through rendering. To solve this problem, surface splatting was proposed [5]. This is a high-quality rendering technique that uses a splat with a Gaussian filter kernel to interpolate information between points. This technique has been developed in various directions. With the increasing programmability of modern graphics processing units (GPUs), many parts of the original algorithm have been delegated to the graphics hardware to improve the rendering speed [6].

Many existing point-based approaches focus on accurate surface reconstructions of point sets considering only geometric information. However, their rendering quality is not high enough to render point-sampled surfaces that are visually sensitive even to fine artifacts such as human faces with a feasible number of points for modern devices.

2.2. Subsampling and Resampling

Sampling is a key process in the reconstruction, rendering, and processing of geometry. Many algorithms have been proposed to reduce the sampling count while keeping the quality of the reconstructions. A common approach is iterative addition and removal of points based on a scoring system that quantifies the change to the reconstruction caused by each operation [7,8]. These methods produce better reconstructions at a higher computational and memory cost. But most fail to reproduce complicated surfaces without artifacts [9].

To promote a good covering of the surface required for high-quality rendering, relaxation methods that move points on the surface iteratively can be utilized. They can be based on inter-point forces [10], variations on Lloyd's algorithm [11,12], advancing front algorithms [13], or splatting-based criteria [14] to distribute the points. Some of these methods can produce both isotropic and anisotropic distributions [15]. However, many of these methods require a good initial distribution, and the quality of the generated samplings are highly dependent on the parameters used. Furthermore, incorporating additional information such as color or facial features cannot be achieved easily. Recently, a spectral algorithm for both reconstruction and distribution purposes has been introduced [9] that overcomes most of these difficulties. We extend this method to include color and salient features for use in facial rendering.

3. BACKGROUND

3.1. Surface Splatting

A splat is a rendering primitive in the form of a disk, which is aligned perpendicular to the surface normal. Each splat consists of an ellipse with a central point, a radius, and a surface color, as well as a normal vector. A point-sampled facial model can be rendered by creating many splats and then interpolating their data where they overlap. The radius of a splat is reduced where points are close together, allowing a detailed rendering of facial features, while other areas with fewer, larger splats appear smooth.

The majority of GPU-accelerated point-based rendering methods employ a three-pass procedure consisting of visibility splatting pass, attribute splatting pass, and shading pass. All these passes are performed by programmable shaders. In the first pass, the surface created by a set of overlapping splats is stored in a depth buffer. In the second pass, surface attributes, such as color values and normal vectors, are accumulated. In the third pass, the image produced during the second pass is normalized, and further per-pixel shading is used to create the final image.

3.2. Spectral Sampling of Point Sets

Spectral sampling [9] is a novel approach for simplification and resampling of surfaces represented as point sets. The essence of this method is to measure the effect of a point on the surface by determining its effect on the Laplace–Beltrami spectrum through an approximation of the heat kernel. Measuring the change in the Laplace–Beltrami spectrum due to a single point is performed by considering the higher dimensional space implied by the heat kernel. This approximation is bound to a kernel in the ambient space and forms a connection to kernel-based reconstruction and rendering.

Efficient rendering requires a sampling density that is dictated by the level of detail. The points should provide an even, isotropic covering of the surface with minimal redundancy. High-quality rendering also requires the avoidance of regularity artifacts. Spectral sampling provides point clouds with these required characteristics. The generated distributions exhibit high-quality blue noise properties, cover the surface well, do not contain regular patterns, and adapt to the geometric or user provided features. These properties allow us to render high-quality facial models without aliasing and smoothing artifacts with much less points.

Using the proposed measure that quantifies the change in the Laplace–Beltrami spectrum due to a single point, we use subsampling and subsequent resampling algorithms [9] to obtain the final sampled point cloud. The measure and, hence, the algorithms depend on a kernel definition given by $k(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||^2}$. For isotropic samplings, $\mathbf{x}$ can be assumed to live in the original 3D space of the point positions. To get samplings of the surface that adapt to the facial feature well, we will use position, normal, color, and facial feature mask information, implying that the sampled manifold is embedded in a higher dimensional space given by the data.

We describe the steps of our sampling and splatting algorithms in the next section.

4. ADAPTIVE SURFACE SPLATTING

Our adaptive surface splatting is designed to be able to realistically render the digital face with a reduced set of points while preserving both geometric and color details (Figure 1). The entire pipeline is illustrated in Figure 2. In the first stage, we assign high weights to important feature areas that need careful handling during sampling and splat optimization. The second stage simplifies the point set using spectral sampling, and the third stage optimizes the shape and size of the splats used for rendering. The last stage renders the final image using the optimized splats.

4.1. Feature Weighting

In order to handle specific feature areas carefully and render the areas with high quality, we can use a masking algorithm based on automatic facial feature detection or interactive freehand drawing.

To detect the eyes and lips, we can apply the AdaBoost cascade classifier [16] to the projected face in the frame buffer. Then, the pixels in the detected feature areas are assigned high weights. The 2D positions of the weighted pixels are reprojected back into the 3D positions on the facial surface. In the case of the eyebrows, we can apply an intensity-based segmentation algorithm to the sub-image above the eyes because the eyebrows can be easily discriminated from facial skin by gray-level intensity.

If a user wants to enhance a specific facial area, a region of interest (ROI) can be drawn on the facial surface interactively. The ROI is projected into the frame buffer, and the pixels in the ROI are masked. We then reproject the weighted pixels back on to the 3D face.

Our current approach is to give the eyes, lips, eyebrows, and facial skin separate labels for weighting, and the weights w are stored for surface sample points as additional information. Hence, for each sample point, the position $\mathbf{p}_i$ normal $\mathbf{n}_i$, color $\mathbf{c}_i$, and weight w_i is stored: $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{n}_i, \mathbf{c}_i, w_i)$.

4.2. Sampling of Point Sets

Realistic rendering of faces requires that the sampling preserves the texture and salient features of the face. Hence, the sampling algorithm needs to allow distributions that adapt to color changes and are sensitive to facial features on the surface. For this purpose, we extend spectral sampling [9] and incorporate color and a feature mask into the sampling process.

In its original form, for adaptive samplings, normals in addition to positions are used in the kernel definition such that $\mathbf{x} = [\mathbf{p}/\sigma_p \ \mathbf{n}/\sigma_n]^T$ [17]. We extend the kernel space to include color components so that it becomes

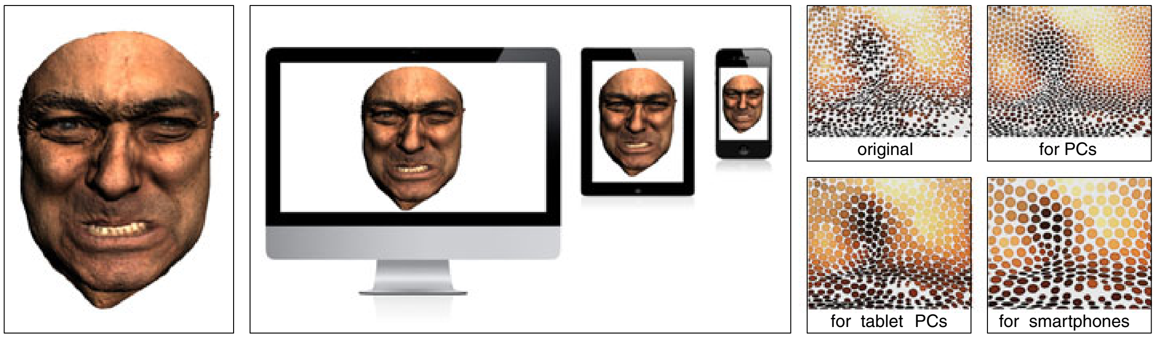


Figure 1. A facial model (left, 1201k points) is rendered by adaptive surface splatting in various devices. Adaptive surface splatting optimizes the facial model for PCs, tablet PCs, and smartphones. The middle figure is rendered according to the rendering performance and resolution of various devices, and the right figures show the splats, which are changed by desired density and a change of color and geometry on the surface.

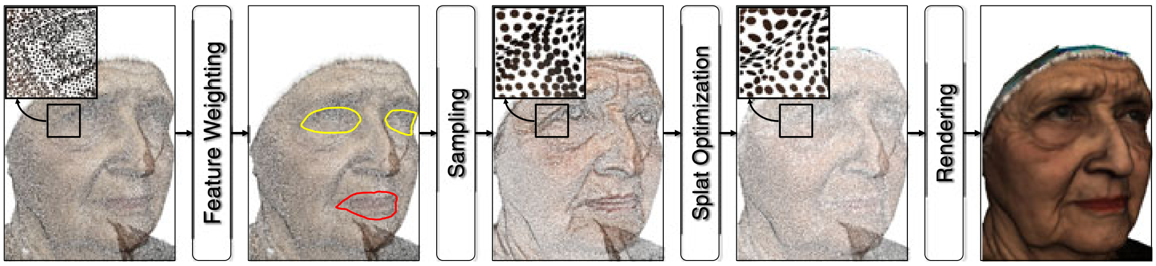


Figure 2. The pipeline of adaptive surface splatting for facial rendering.

$\mathbf{x} = [\mathbf{p}/\sigma_p\ \mathbf{n}/\sigma_n\ \mathbf{c}/\sigma_c]^T$, where σ_c controls the sensitivity to color changes in the model.

Note that we only have color values for the initial points, but values in space are needed for resampling. Following the original derivation of the adaptive kernel definition from robust statistics [17], we use robust local regression on colors to get a value $\mathbf{c}(\mathbf{x})$ for any point $\mathbf{x}$ using the following iterative formula:

$$\mathbf{c}^{k+1} = \frac{\sum_i \mathbf{c}_i k(\mathbf{x}^k, \mathbf{x}_i)}{\sum_i k(\mathbf{x}^k, \mathbf{x}_i)} \qquad (1)$$

where $\mathbf{x}^k$ includes the current estimate of the color $\mathbf{c}^k$. Starting from the usual kernel approximation $\sigma_c = \infty$, iteration continues until convergence. This gives us a robust color estimate that accurately preserves discontinuities in colors [17]. After computing the new color $\mathbf{c}$, we then form the vector $\mathbf{x}$ and proceed with resampling [9].

The final component of our sampling algorithm is utilizing the facial mask for preserving perceptually important features. Denoting the facial mask with $m(\mathbf{x})$ for the point $\mathbf{x}$, the sampling density can be adjusted by weighting the parameter σ_p that controls the spatial extent of the kernel. Using lower values for this parameter will result in finer details and more sampled to be placed. Hence, we use the following final form $\mathbf{x} = [\mathbf{p}/(\sigma_p w)\ \ \mathbf{n}/\sigma_n\ \ \mathbf{c}/\sigma_c]^T$. The mask weights w depend on the region of the face and are smaller for the important facial features of eyes, lips, and eyebrows. We experimented with different settings, and the simple selection of $w = 1$ for skin and $w = 0.5$ for facial feature areas works well in practice for generating perceptually correct renderings.

4.3. Splat Optimization for Rendering

The next stage in our pipeline after sampling is splatting the generated point cloud data. Although we get an adaptively sampled point set after the sampling stage, directly rendering it with circular splats will still produce over-smoothed renderings. To render accurate images, the splat sizes and shapes should be adapted to the point distributions generated in the sampling stage.

Ideally, splat shapes should follow the kernel shapes at each sample point. However, the kernels used can have very complicated shapes depending on the attributes at the neighboring sample points. Hence, directly translating kernel shapes into splat shapes results in an unfeasible complexity for the splatting stage with many parameters per splat, and the resulting splats cannot be easily rendered using existing methods.

Refraining from a full shape adaptation, we limit the class of splat shapes to those of ellipses. Thus, instead of using regular disks with a constant radius for the splats, we allow elliptical shapes with different extents for each axis and sample point as shown in Figure 3. Each splat is an elliptical disk centered around the sample point position $\mathbf{p}$, oriented orthogonal to the normal $\mathbf{n}$, and has the axes $\mathbf{u}$ and $\mathbf{v}$ and the splat color $\mathbf{c}$. The lengths of $\mathbf{u}$ and $\mathbf{v}$ define the extent of the ellipse in the corresponding directions.

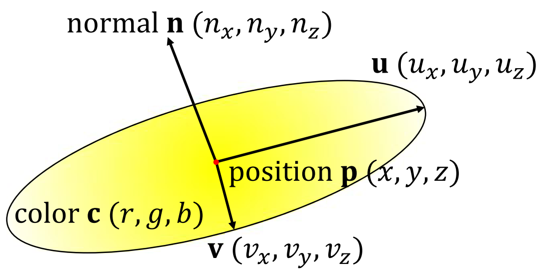

normal $\mathbf{n}$ (n_x, n_y, n_z)
$\mathbf{u}$ (u_x, u_y, u_z)
position $\mathbf{p}$ (x, y, z)
color $\mathbf{c}$ (r, g, b)
$\mathbf{v}$ (v_x, v_y, v_z)

Figure 3. The shape of an elliptical splat.

Defining the splat shapes in this way reduces the splat optimization stage to finding the parameters $\mathbf{u}$ and $\mathbf{v}$ for each sample point. Intuitively, the disks should capture a coarse version of the kernel shapes on the tangent plane, which depends on the distribution of the samples $\mathbf{x}_i$ around a given sample point $\mathbf{x}$. To capture the main axes of the kernel shape, we use a weighted combination of projected direction vectors on the tangent plane.

First, we compute the projection of the vectors $\mathbf{x} - \mathbf{x}_i$ onto the tangent plane: $\mathbf{P}(\mathbf{x} - \mathbf{x}_i)$. The matrix $\mathbf{P}$ zeroes out all the components that are not in the spatial domain (i.e., normal and color) and projects the position $\mathbf{p}$ onto the tangent plane such that the projection reduces to $\mathbf{y}_i = (\mathbf{I} - \mathbf{n}\mathbf{n}^T)(\mathbf{p} - \mathbf{p}_i)$. Using these projected vectors and weighting by the kernels, we get the following covariance matrix:

$$\mathbf{C} = \sum_i \mathbf{y}_i \mathbf{y}_i^T k(\mathbf{x}, \mathbf{x}_i) \qquad (2)$$

This covariance matrix has a zero eigenvalue and two other positive eigenvalues α_1 and α_2 and corresponding eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$. These eigenvalues capture the extent of the shape of the kernel in the corresponding directions.

Note that the eigenvalues represent only the relative sizes and hence computing the absolute shapes of the splats should be performed by also considering appropriate scaling. We use a local scaling term that adapts to the relative placement of splats and also a global scaling term that adjusts the overall smoothness of the rendering.

Second, the scale factor of the shape of each splat is determined in a locally adaptive way. For the sample $\mathbf{x}$, the distance to the closest sample $\mathbf{x}^c$ is computed. Then, the ratio of this distance to the average of all closest distances is used as the local scale factor h_l, which is given by the following equation:

$$h_l = \frac{\|\mathbf{x} - \mathbf{x}^c\|}{n^{-1} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_i^c\|} \qquad (3)$$

Using this factor along with the global scale factor h_g, we can finally set $\mathbf{u} = h_l h_g \alpha_1^2 \mathbf{e}_1$ and $\mathbf{v} = h_l h_g \alpha_2^2 \mathbf{e}_2$.

4.4. Final Rendering

Although we can guarantee visually continuous appearance of the displayed objects by creating many splats or increasing the size of splats, the number of processed splats and overlapping areas are also increased. The increment of overlapping not only slows down the rendering process significantly but also causes blurring artifacts.

Our adaptive surface splatting method is designed to cover more surface area while keeping the overlap as small as possible. This characteristic essentially comes from the use of elliptical splats with different shapes and sizes. It is especially beneficial when rendering small-scale feature areas with abrupt color changes.

In order to maintain the benefit, we use a hole-filling algorithm in image space instead of increasing the size of splats excessively in object space to attempt to guarantee a hole-free representation. Covering all point samples by splats in object space does not guarantee a hole-free representation because holes can appear in-between the point samples. If the splat size is increased to fill the hole, then the surface splatting process is slowed because the number of the fragments projected on the screen space is increased. Moreover, the overlapping areas lead to a smoothly blurred image.

The hole-filling algorithm is an additional process based on the three-pass surface splatting on GPUs [6]. The hole-filling process is performed in the last pass (i.e., normalization pass). This pass is independent of the scene complexity and is negligible compared with the rendering time of the two previous splatting passes (i.e., visibility splatting and attribute splatting passes).

In the last pass, we can obtain results of the attribute splatting, which are images of albedo color and normal map on the frame buffer object (FBO). If the hole size s_h is sufficiently small in these images (below a threshold value), the color and normal values of the hole are interpolated by the depth-based hole-filling algorithm. In this process, FBO is initialized to a specific value (e.g., 0.0) before the first pass, and areas that are not filled by splats still have the initial value after the second pass. Therefore, the hole are determined as the area that are inside the boundary of the rendered model with FBO set to the initial value. A pseudocode of the algorithm is given in Algorithm 1.

5. RESULTS

We have applied our adaptive surface splatting method to a wide range of different facial data sets obtained by the face-scan team at ETH Zürich [2] to demonstrate the quality and performance. We used facial data sets of people in Asia, Europe, and Africa, with the aim of assessing how differences in each facial feature depend on skin color and the extent of wrinkling.

5.1. Sampling of Facial Data Sets

To control the extent to which geometric features are preserved, we use the parameter σ_n. The top row of Figure 4 shows the sampling results of specific facial areas with fine lines or wrinkles. To demonstrate the effect of our sampling algorithm for detailed facial geometry, we tested the

Algorithm 1 Depth based hole-filling

```
Input: rendered images of color $C$, normal $N$, depth $D$,
       the threshold of depth $th_d$,
       the width $w$ and height $h$ of the images,
       the hole-filling filter size $s_h$.
Output: hole-filled information of color $C^o$, normal $N^o$.

for $((i,j) = (0,0)$ to $(w,h))$
  if ($D_{ij}$ is 0.0 and isHoleInsideModel$(i,j)$)
    $C^o_{ij} = N^o_{ij} = g_s = 0.0$
    for each ($k, l$ with $|k-i| < s_h, |l-j| < s_h$)
      if ($|D_{kl} - D_{ij}| < th_d$)
        $C^o_{ij}$ += $C_{kl}\ g(i-k, j-l)$
        $N^o_{ij}$ += $N_{kl}\ g(i-k, j-l)$
        $g_s$ += $g(i-k, j-l)$
      end
    end
    $C^o_{ij}$ /= $g_s$, $N^o_{ij}$ /= $g_s$
  end
end
```

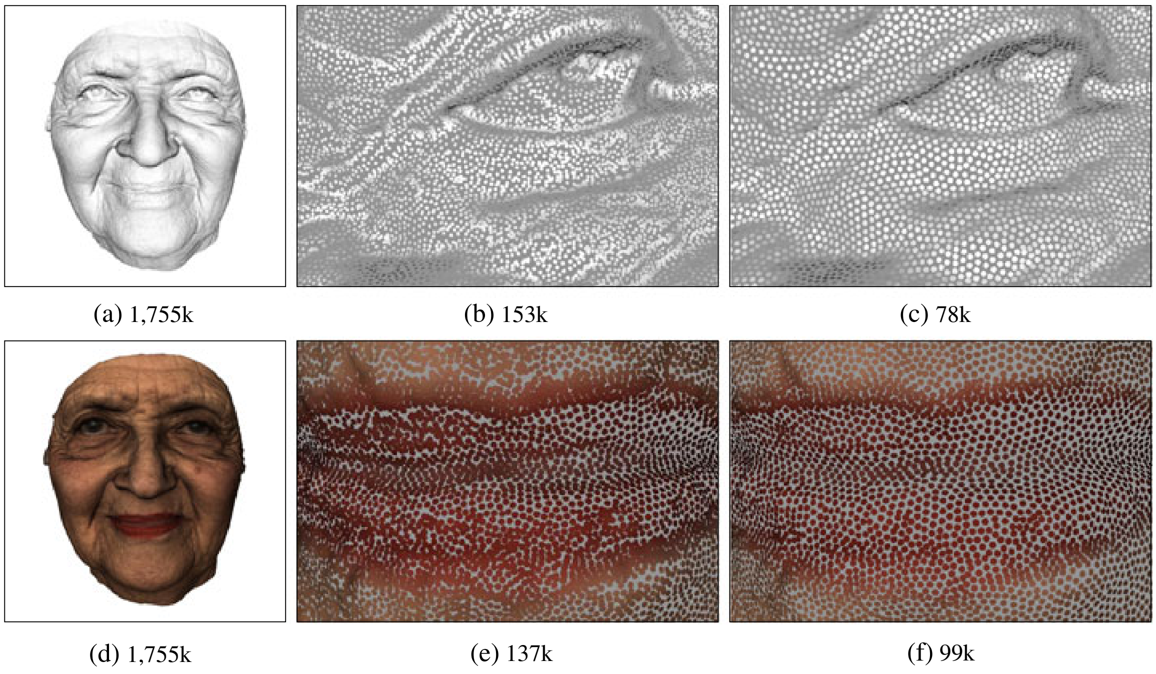


Figure 4. (a) original, (b) $\sigma_n = 0.3$, (c) $\sigma_n = 0.9$, (d) original, (e) $\sigma_c = 0.05$, (f) $\sigma_c = 0.1$.

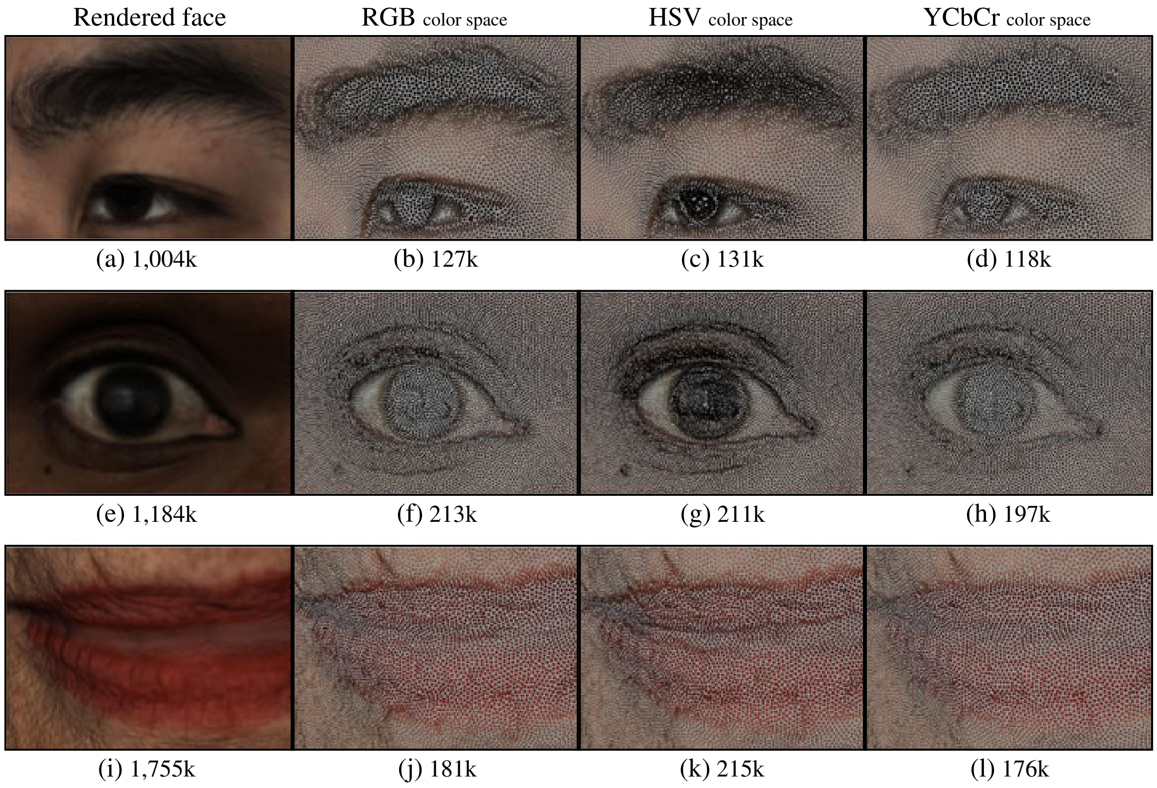


Figure 5. Comparison of sampling results using different skin color groups and color spaces.

algorithm on facial geometric data without color information. Figures 4(a–c) show how the distribution of point samples varies by σ_n ($\sigma_p = 0.01$, $\sigma_c = 0.9$). We can see fine lines and wrinkles on the faces with less than 95% of the point samples. To determine the extent to which color features are preserved after sampling, the value of σ_c can be adjusted by a user. Figures 4(d–f) show the sampling results of the facial features of non-skin color with different σ_c values ($\sigma_p = 0.01$, $\sigma_n = 0.9$). Even if more than 95% of point samples were reduced, the small-scale color features, such as lip lines, were preserved well.

Next, we ran experiments to discover whether our adaptive sampling algorithm is affected by different skin colors or color spaces. To demonstrate the robustness of our algorithm to skin colors, we tested facial data of different human races. Figure 5 shows the results of our algorithm with Asian, African, and European facial data with the same parameter values ($\sigma_p = 0.006$, $\sigma_n = 0.9$, $\sigma_c = 0.1$). Regardless of the skin colors, our algorithm is effective to preserve non-skin color features such as eyes and lips.

Additionally, three different color spaces were compared. Different color spaces usually possess different color characteristics, which can be applied for different visual tasks, such as detection and recognition. As shown in Figures 5(c, g, and k), the HSV (hue, saturation, value) color space outperforms the other color spaces in detecting facial features, such as pupils, lip lines, and eyebrows.

5.2. Facial Rendering

To test usage of the massive facial data we have on tablet PCs or smartphones, we rendered the facial data using our adaptive surface splatting with different number of splats. Figure 6 shows that our method preserves small-scale facial features even after drastic simplification. Figure 6(a) is the rendering result using the original dataset. Figures 6(b, c, and d) are the results after 50%, 81%, and 97% simplification, respectively. Figure 7 illustrates the original point distribution and the shape, size, and distribution of splats after splat optimization.

In order to assess the quality improvement, we compare our method with different surface splatting methods in Figure 8. Figure 8(a) is the rendering result with $1004k$

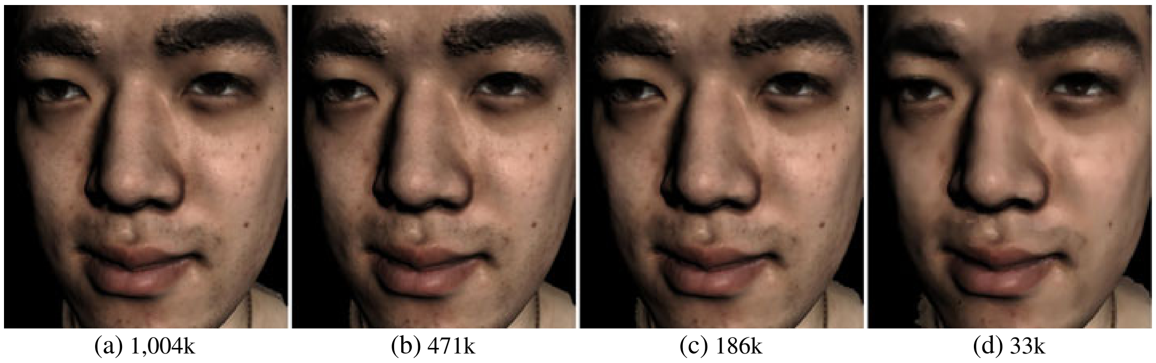


Figure 6. Adaptive surface splatting of facial data.

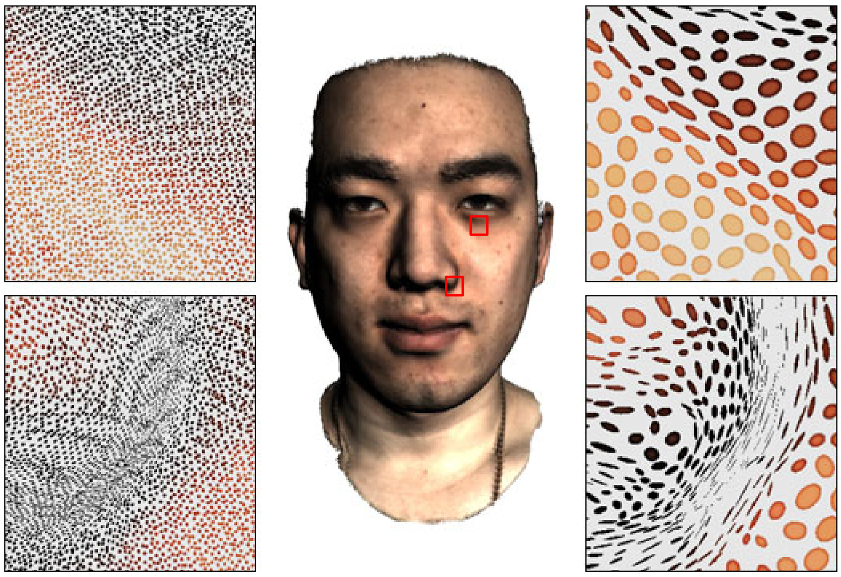


Figure 7. Comparison of the original point distribution (left) and the splat distribution (right).

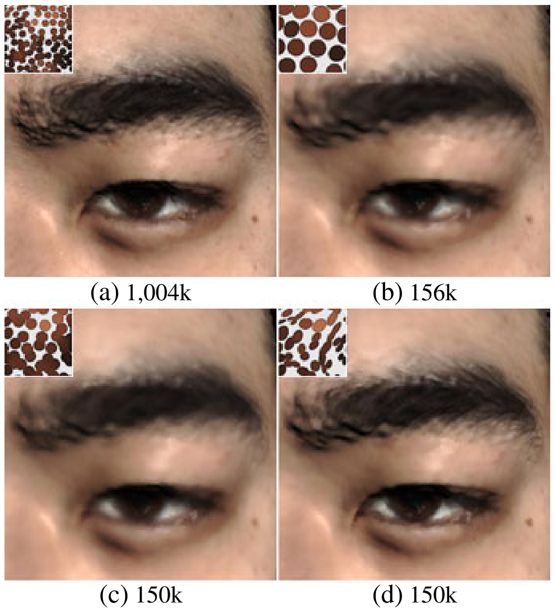


Figure 8. Comparison of surface splatting methods: (a) without simplification, (b) uniform sampling, (c) extended spectral sampling, and (d) our adaptive surface splatting.

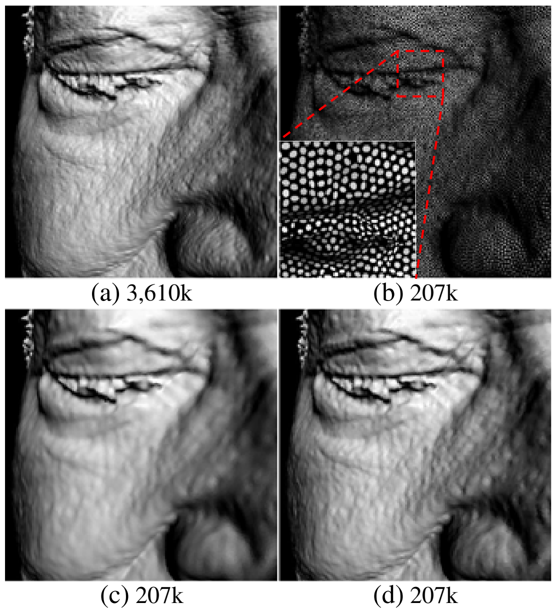


Figure 9. Comparison of rendering quality: (a) original, (b) points of a simplified face, (c) typical surface splatting, and (d) adaptive surface splatting using hole filling.

splats without simplification. Figure 8(b) is the result of a typical surface splatting technique [6] using uniform sampling after 85% simplification. Figure 8(c) is the result of the same splatting method using extended spectral sampling considering geometry and color. Figure 8(d) is the result of our adaptive surface splatting after 85% simplification. Compared with other renderings with simplified datasets, our result preserves detailed facial features such as eyebrows better.

To demonstrate that our final rendering stage using hole-filling improves the rendering quality with less blurring artifacts, we compare our method with standard surface splatting [6] using facial data without color information (Figure 9). Compared with Figure 9(c), our result, Figure 9(d), preserves detailed geometry, creating a realistic face.

Finally, Table I shows a performance comparison according to the number of points on various devices. As the number of points are reduced, rendering the models on smaller devices become feasible.

5.3. Application to Subsurface Scattering

Recently, a surface splatting method for simulating scattering of light in skin has been introduced [18,19]. This method assumes that the diffusion of light is isotropic; therefore, its effect at a splat can be expressed as a Gaussian distribution applied to the radius of the splat. A diffusion profile is composed of six layers of increasing radius using a sum-of-Gaussian technique [20]. Our adaptive sampling can be used to optimize the number of splats for each layer, resulting in an overall speedup.

Table I. Rendering performance on various devices.

Devices	Number of points			
	$1004k$	$471k$	$186k$	$33k$
PC (fps)	28	76	120	168
iPad 2 (fps)	2	8	14	18
iPhone 3GS (fps)	1	2	4	8

Figure 10 shows the comparison of the rendered images without and with subsurface scattering. Figure 10(a) is the result of the elliptical weighted average surface splatting [6] using a single layer and is hard and dry-looking. Figure 10(b) is the rendered image by our surface splatting using six layers of optimized splats to simulate the light scattering process that takes places inside translucent materials. This subsurface scattering effect gives skin its soft appearance and creates the semi-translucent appearance of the realistic human skin. The surface splatting with six-layer non-optimized splats ($2361k$) was rendered at 20 fps, and Figure 10(b) with six-layer optimized splats ($1831k$) was rendered at 27 fps on an Intel Core 2 Quad CPU Q6600 2.4-GHz PC with an nVidia Quadro FX 3800 graphics card. Hence, our adaptive sampling can improve the rendering speed by optimizing the number of splats.

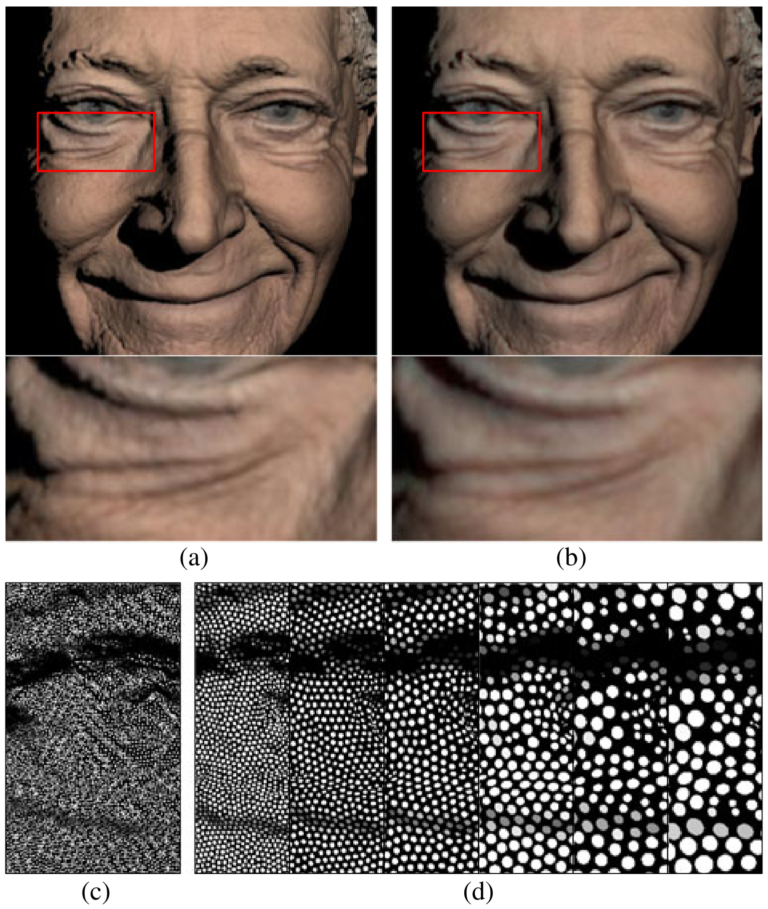


Figure 10. Comparison of rendering results without and with subsurface scattering: (a) without subsurface scattering, (b) adaptive surface splatting with subsurface scattering, (c) original points distribution, and (d) six layers with optimized splats.

6. CONCLUSIONS

We developed an adaptive approach to simplify massive facial data and render it with high quality. The whole process directly works on point samples without any connectivity information. This allows us to render the facial data without any surface polygonization and parameterization for texture mapping, resulting in a much simpler process for high-density point sets. Compared with previous works, our adaptive surface splatting method shows a higher quality of results by guiding the simplification and splat optimization processes with face specific details. Our method is particularly effective at preserving the perceptually important facial features such as eyes, fine lines, wrinkles, or lip lines.

For future work, we would like to port our sampling and splat optimization algorithms to GPU so that the point density of the entire face can be adjusted in real time. We would also like to investigate incorporating other perceptual measures into our algorithms.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Culture, Sports and Tourism and KOCCA in the CT R&D Program 2011. This work was also supported by the National Research Foundation of Korea (2010-0019373 and F01-2009-000-10029-0) and the Swiss National Science Foundation (200021-130379 and 200021-132438/1).

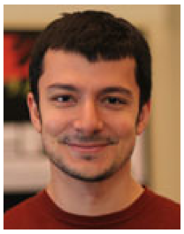
REFERENCES

1. Calvo MG, Nummenmaa L. Detection of emotional faces: salient physical features guide effective visual search. *Journal of Experimental Psychology* March 2008; **137**(3): 471–494.
2. Beeler T, Bickel B, Beardsley P, Sumner B, Gross M. High-quality single-shot capture of facial geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH 2010)* July 2010; **29**(4): 40:1–40:9.

3. Weyrich T, Matusik W, Pfister H, Bickel B, Donner C, Tu C, McAndless J, Lee J, Ngan A, Jensen HW, Gross M. Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)* July 2006; **25**(3): 1013–1024.
4. Ghosh A, Hawkins T, Peers P, Frederiksen S, Debevec P. Practical modeling and acquisition of layered facial reflectance. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2008)* December 2008; **27**(5): 139:1–139:10.
5. Zwicker M, Pfister H, van Baar J, Gross M. Surface splatting, In *Proc. SIGGRAPH 2001*, Los Angeles, USA, August 2001; 371–378.
6. Botsch M, Hornung A, Zwicker M, Kobbelt L. High-quality surface splatting on today's GPUs, In *Symposium on point-based graphics 2005*, Stony Brook, USA, June 2005; 17–24.
7. Garland M, Heckbert PS. Surface simplification using quadric error metrics, In *Proc. SIGGRAPH 1997*, Los Angeles, USA, August 1997; 209–216.
8. Alexa M, Behr J, Cohen-Or D, Fleishman S, Levin D, Silva CT. Point set surfaces, In *Proc. IEEE Visualization 2001,* VIS 2001, San Diego, USA, October 2001; 21–28.
9. Öztireli AC, Alexa M, Gross M. Spectral sampling of manifolds. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2010)* December 2010; **29**(6): 168:1–168:8.
10. Witkin AP, Heckbert PS. Using particles to sample and control implicit surfaces, In *Proc. SIGGRAPH 1994*, Orlando, USA, August 1994; 269–277.
11. Yan DM, Lévy B, Liu Y, Sun F, Wang W. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum (Proc. Eurographics SGP 2009)* July 2009; **28**(5): 1445–1454.
12. Valette S, Chasseryand JM, Prost R. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics* March 2008; **14**(2): 369–381.
13. Schreiner JM, Scheidegger CE, Fleishman S, Silva CT. Direct (re)meshing for efficient surface processing. *Computer Graphics Forum (Proc. Eurographics 2006)* September 2006; **25**(3): 527–536.
14. Wu J, Kobbelt L. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum (Proc. Eurographics 2004)* September 2004; **23**(3): 643–652.
15. Li H, Wei LY, Sander PV, Fu CW. Anisotropic blue noise sampling. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2010)* December 2010; **29**(6): 167:1–167:12.
16. Viola P, Jones MJ. Robust real-time face detection. *International Journal of Computer Vision* May 2004; **57**(2): 137–154.
17. Öztireli AC, Guennebaud G, Gross M. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum (Proc. Eurographics 2009)* April 2009; **28**(2): 493–501.
18. Kim HJ, Bickel B, Gross M, Choi SM. Subsurface scattering using splat-based diffusion in point-based rendering. *SCIENCE CHINA Information Sciences* May 2010; **53**(5): 911–919.
19. Choi SM. Subsurface scattering in point-based rendering. *Chinese Science Bulletin 2010* August 2010; **55**(23): 2598–2598.
20. d'Eon E, Luebke D, Enderton E. Efficient rendering of human skin, In *Rendering Techniques 2007: 18th Eurographics Symposium on Rendering*, Grenoble, France, June 2007; 147–158.

AUTHORS' BIOGRAPHIES

Hyeon-Joong Kim is a PhD student in the Department of Computer Science and Engineering at Sejong University, Seoul, Korea. He received his BS and MS degrees in Computer Science and Engineering from Sejong University in 2007 and 2009, respectively. His current research interests include computer graphics, virtual reality, point-based graphics, and realistic facial rendering.



A. Cengiz Oztireli is a PhD student at ETH Zürich in the Computer Graphics Laboratory. His research interests include manifold reconstruction and sampling, meshfree methods, and sketch-based modeling. He earned his Master of Science degree in Visual Computing from ETH Zürich in 2008 and Bachelor degrees in Computer Engineering, and Electrical & Electronics Engineering from Koc University, Istanbul in 2006. He has received several awards and research grants including an SNF research grant and best student paper award in Eurographics 2009.

Markus Gross is a professor of Computer Science at the Swiss Federal Institute of Technology Zürich (ETH), head of the Computer Graphics Laboratory, and the Director of Disney Research, Zürich. He joined the ETH Computer Science faculty in 1994. His research interests include physically based modeling, computer animation, immersive displays, and video technology. Before joining Disney, Markus was director of the Institute of Computational Sciences at ETH. He received a Master of Science in Electrical and Computer Engineering and a PhD in Computer Graphics and Image Analysis, both from Saarland University in Germany in 1986 and 1989. Markus serves on the boards of numerous international research institutes, societies, and governmental organizations. He received the Technical Achievement Award from EUROGRAPHICS in 2010 and the Swiss ICT Champions Award in 2011. He is a fellow of the EUROGRAPHICS Association and a member of the German Academy of Sciences Leopoldina. Prior to his involvement in Disney Research, he cofounded Cyfex AG, Novodex AG, LiberoVision AG, and Dybuster AG.

Soo-Mi Choi is currently an Associate Professor in the Department of Computer Engineering at Sejong University, Seoul, Korea. She received her BS, MS, and PhD degrees in Computer Science and Engineering from Ewha University of Korea in 1993, 1995, and 2001, respectively. Between June 1998 and December 1998, she was with the Fraunhofer Institute for Computer Graphics Research (IGD) of Germany as a visiting researcher. After she received her PhD, she joined the Center for Computer Graphics and Virtual Reality (CCGVR) at Ewha University as a Research Professor. In 2002, she became a faculty member in the Department of Computer Engineering at Sejong University. From September 2008 to August 2009, she was a visiting scholar at the CG Lab of ETH Zürich in Switzerland. She serves as a member of program committees of many conferences on computer graphics and human–computer interaction. Her current research interests include computer graphics, virtual reality, human–computer interaction, and ubiquitous computing.