Pacific Graphics 2018
H. Fu, A. Ghosh, and J. Kopf
(Guest Editors)

Volume 37 (2018), Number 7

Instant Stippling on 3D Scenes

Lei Ma[1,3], Jianwei Guo[2,3], Dong-Ming Yan[2,3], Hanqiu Sun[4], Yanyun Chen[1,3†]

[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences;
[2]NLPR, Institute of Automation, Chinese Academy of Sciences; [3]University of Chinese Academy of Sciences; [4]Chinese University of Hong Kong
E-mails: malei@outlook.com; jianwei.guo@nlpr.ia.ac.cn, yandongming@gmail.com, hanqiu@cse.cuhk.edu.hk, chenyy@ios.ac.cn

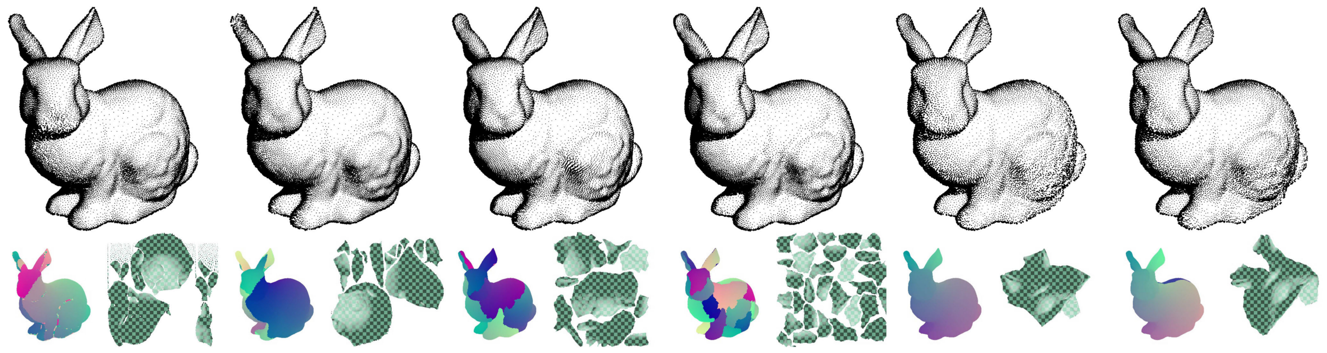

Figure 1: *Our stippling results with different surface parameterization methods. From left to right: LSCM [LPRM02] with angle* 15°*, LSCM with angle* 60°*, IsoChart [ZSGS04] with 9 charts, IsoChart with 30 charts, BFF [SC17] with 10 cones, and BFF with 18 cones. The UV maps are shown on the bottom and the corresponding stippling results are shown on the top.*

Abstract

In this paper, we present a novel real-time approach to generate high-quality stippling on 3D scenes. The proposed method is built on a precomputed 2D sample sequence called incremental Voronoi set *with blue-noise properties. A rejection sampling scheme is then applied to achieve tone reproduction, by thresholding the sample indices proportional to the inverse target tonal value to produce a suitable stipple density. Our approach is suitable for stippling large-scale or even dynamic scenes because the thresholding of individual stipples is trivially parallelizable. In addition, the static nature of the underlying sequence benefits the frame-to-frame coherence of the stippling. Finally, we propose an extension that supports stipples of varying sizes and tonal values, leading to smoother spatial and temporal transitions. Experimental results reveal that the temporal coherence and real-time performance of our approach are superior to those of previous approaches.*

CCS Concepts

•***Computing methodologies*** $\rightarrow$ ***Computer graphics; Non-photorealistic rendering; Image processing;*** *Texturing; Appearance and texture representations;*

1. Introduction

Stippling is a well-known *non-photorealistic rendering* (NPR) technique that uses only plain dots to represent the tone and shading of images or 3D scenes. Stippling style has been used in various applications, ranging from scientific visualization to informative illustration.

Stippling artworks often contain a large number of stipples to capture the subtle transition of tones. This process is tedious even for a skillful artist because he or she has to spend several hours or even days to paint the picture. Manual stippling for dynamic 3D scenes is even more challenging because keeping the temporal coherence for a massive number of stipples is almost impossible. Various techniques have been developed to produce high-quality 2D stippling effects for images and facilitate this process [MARI17]. Some of them have been generalized to 3D surfaces [YNZC05], volumes [LMT*03], or frame-coherent animation sequences [PFS03, VBTS07]. However, these 3D stippling methods rely on either using stipple texture images or precomputing vertices of different densities. Therefore, the qualities of resulting stipple distributions are either far from uniformness or lack of randomness, compared to their 2D counterparts.

† corresponding author

In this paper, we propose a novel method to generate high-quality stippling effect for 3D scenes in real-time. The key idea is to map a certain number of stipples into the given 3D surface with the guidance of texture coordinates. Dense stipple candidates are initially placed on surfaces through parametric spaces. A 2D incremental sequence called incremental Voronoi set (IVS) [MCQS18] is used for producing the stipple candidates. A carefully designed thresholding strategy is utilized to produce the stippling results. Shading and parametrization distortions are also considered in this process. As a result, the proposed technique exhibits both high-quality stipple distribution and real-time efficiency and is intrinsically temporal coherent, which is essential in many 3D applications.

Most existing digital stippling methods are restricted to only use uniform-sized and mono-black stipples. However, as a basic painting technique, stippling skills should not be limited on these aspects. The original stippling for engraving purpose has adopted varying stipple sizes to create smooth shading transitions. Modern artists also use pens of multiple tones for this similar effect. To mimic this effect, we further introduce new techniques to generate stipples of varying radius, and multi-tone levels for 3D scenes (as carried out in 2D by [MCQS18]).Results of these techniques show visual satisfaction and maintain the important temporal coherence. More importantly, these two extensions introduce very limited computation cost without detriment to the performance.

In summary, our main contributions are as follows:

- a high-quality stippling technique for dynamic 3D scenes based on a precomputed IVS with blue-noise properties;
- parallelization of the proposed technique to improve the performance based on the thresholding of individual stipple;
- extension of the proposed method to generate varying radius and multi-tone stippling to improve the rendering expressiveness.

2. Related Work

The stippling effect can be produced via blue-noise sampling, which ensures both the randomness and uniformness of the point distribution [LD08, YGW*15]. The three main categories of 2D stippling techniques are maximal Poisson-disk sampling and its variations, optimization-based approach, and precomputed tiles. The former two types of work can be generalized to 3D surfaces or even high dimensions directly, but the performance is far from real-time. The tile-based approach could achieve real-time performance by scarifying the sampling quality, but the extension to 3D surface is not straightforward. In this section, we briefly discuss the representative stippling techniques, with a focus on surface stippling.

2.1. 2D stippling

Optimization-based approaches include *centroidal Voronoi tessellation* (CVT) [DFG99] and its variation, namely, *capacity-constrained Voronoi tessellation* (CCVT) [BSD09], and *capacity-constrained Delaunay triangulation* (CCDT) [XLGG11]. Starting from an initial randomized sampling, CVT-based approaches iteratively move each sample point to the centroid of its Voronoi cell, and update the Voronoi diagram accordingly [DHvOS00, Sec02, DSZ17]. Although CVT stippling could obtain the importance information of input images, it tends to introduce regular patterns that may cause visual aliasing, because the global minimum of the CVT energy is the regular hexagonal pattern. To improve the visual aesthetics, the CCVT-based methods enforce the capacity-constraint in addition to CVT energy, which can be seen as assigning the same amount of ink to each stipple [BSD09, dGBOD12, CYC*12, AHD15]. Other optimization-based approaches include kernel density model [Fat11], simple push-pull algorithm [AGY*17]. However, due to their optimization nature, these methods cannot achieve real-time performance.

Another way to generate stippling pattern is through Poisson-disk sampling [DW85, Coo86, Jon06, WCE07, Wei08, GM09, Wei10, EPM*11, YW13] and its variations [Yuk15]. Poisson-disk stippling can be generated in one pass without iterative optimization. Although this sampling has better randomness than CVT-like methods have, it still cannot achieve real-time performance even with GPU acceleration for large-scale scenes.

Tile-based approaches are efficient for 2D stippling, which use precomputed tiles for online generating points, *e.g.*, recursive Wang-tiles [KCODL06], pentagon patterns [ODJ04], polyominoes [Ost07], and hexagon tiles [WPC*14]. These tile-based approaches achieve real-time performance by scarifying the blue-noise property to some extent. In contrast to tile-based methods, Ahmed et al. developed other efficient look-up methods by encoding the point sequence using strings [AHD15, ANHD17] or low-discrepancy sets [APC*16]. More recently, Ma et al. [MCQS18] proposed a novel method to generate 2D stippling in real time. Their technique is based on a precomputed sample sequence called IVS. The main advantage of IVS over other 2D sample sequences is that it is incremental and highly parallelizable while maintaining blue-noise properties.

Apart from the aforementioned popular techniques, other conventional methods for 2D stippling such as screening-based methods exist. These methods operate on a discrete image pixel spatially to determine whether a microdot should receive ink or not [Uli87]. They produce results on a discrete image and are highly parallelizable. Mitsa *et al.* [MP92] and Ulichney [Uli93] proposed using blue-noise dithering masks over the Bayer matrix [Bay73]. A comprehensive discussion of this topic is out of the scope of this paper. For more details, recent surveys on 2D stippling and half-toning techniques are found in [DI13, MARI17].

2.2. 3D stippling

The optimization-based and Poisson-disk sampling related stippling approaches can be naturally generalized to 3D surfaces [CJW*09, BWWM10, CCS12, YW13, GYJZ15, MIPS14, Yuk15, QGY*16] or even higher dimensions [Bri07, EMP*12, MEA*18]. However, certain drawbacks in their 2D counterpart such as low performance can be observed.

To improve the efficiency of stippling on surfaces, various acceleration techniques have been developed. In the study of Vanderhaeghe *et al.* [VBTS07], they first generated 2D stipples and then projected these stipples onto animated 3D surface in the image space, with an additional epenthesis on keeping the coherence between animated frames. Baer *et al.* [BTBP07] introduced a real-time texture-based stippling method that uses a special polycube representation to minimize the texture distortions. Their stippling illustrations are produced by mixing the texture value between different dot textures on different min-map levels, similar to that of [PHWF01]. Krãijger and Westermann [KW07] introduced 3D noise textures for generating stippling effect on surfaces. Although both image-based and texture-based approaches could achieve real-time performance, the visual aesthetics might be low because the stippling dots are usually texturing mapping results from several continuous min-maps.

The realism of stippling on surface could be improved by performing direct computation in object space. Pastor *et al.* [PFS03] used mesh preprocessing to generate stipples for 3D surfaces. They first assigned a stipple to each vertex of the mesh, then randomly perturbed the stipple positions to avoid artifacts and to adjust the density of stipples using mesh subdivision and the preprocessed progressive mesh. Similarly, Sousa *et al.* [SFWS03] marked to each edge and relied on dense meshes and adjusted each mark accordingly. Pastor *et al.* [PS04] further extended Sousa *et al.*'s idea by organizing the points hierarchically using a graph. Our proposed technique does not require any mesh preprocessing compared with these methods.

One close work to ours is that of Yuan *et al.* [YNZC05], who proposed a texture space NPR method to achieve several different rendering styles. They first computed a geometric image of the input mesh using conformal parametrization. Then they generated the stipples on image space using the important sampling technique [ODJ04]. However, using a single geometry image to represent the mesh surface might cause large distortions. A key difference between our approach and their approach [YNZC05] is that we parameterize the mesh surface with multiple patches instead of only one, and our algorithm is robust to the parametrization methods. Another improvement we made in this paper is that we use an incremental sequence for thresholding rather than 2D tile-based approach. The main advantage is that, based on the proposed incremental sequence, thresholding to adaptive control is trivially parallelizable, which is very friendly for GPU rendering. Our method also preserves blue-noise property and has no limitation on target sampling data. More importantly, its incremental nature allows users to change the number of stipples as they desire. This is suitable for our instant stippling on 3D surfaces.

Other NPR techniques for 3D surfaces include real-time hatching [PHWF01, SBB16], surface decoration [WLY*16] and streamline visualization [PZ07]. More details can be found in [LVPI18].

3. Stippling for 3D dynamic Scenes

The main difficulty of image stippling is obtaining a random uniform distribution of stipples and capturing the local tone carefully. Stippling for 3D scenes is more challenging because keeping temporal coherent for numerous stipples becomes another unavoidable issue. Therefore, although real-time NPR techniques for dynamic scenes have been explored for more than a decade [PHWF01, PFS03, YNZC05, SBB16], stippling being one of the most conventional sketching techniques is still considered one of the good solutions for dynamic 3D scenes.

We assume the following criteria for 3D surfaces stippling:

1. As a stylized rendering technique, every individual frame must appear to be a stippling artwork.
2. For dynamic scenes, the stipples should *anchor* and move with objects instead of the screen space.
3. Stipples are discrete primitives; therefore, flicker is unavoidable, but it should be reduced to an acceptable level.

By these, we have the following factors to maintain the *temporal consistency* of the algorithm design:

- Orientation: stipples must all be facing to the camera.
- Position: all stipples must anchor onto the surface of 3D objects.
- Incrementality: the number, shapes, or tones of local stipples must change incrementally for consequent frames to eliminate the flickers.

To this end, a sample sequence called IVS is proposed to create stipples on 3D surfaces.

3.1. The IVS Sequence

The IVS is a sequence of Voronoi points starts from a given number of random locations on a toroidal domain and incrementally adds consequent points until infinity [MCQS18]. The IVS sequence has an interesting property; that is, the sequence index of an individual sample during the generation is approximately inversely proportional to its Voronoi area during generation. Whenever the sequence is terminated, the generated samples are almost evenly distributed on the sampled domain and show good blue-noise properties. The surrounding area a_i of the i^{th} sample is $a_i \approx A_0/i$, where A_0 is the total area of the stippling domain specified by user.

3.2. Surface Stippling using IVS Sequences

Our basic idea is to provide an IVS sequence of overwhelming length, find the unique location on the target surfaces for each sample, and then reject many of the samples by several thresholding processes to obtain the final distributions. These thresholding processes cover the compensate of the area distortion of parameterizations, the change of the appearance areas after perspective projection onto the screen, and the elimination of excessive samples (black stipples) at the lit region to match the shading appearance after lighting computation. The complete algorithm is illustrated in Fig. 2.

3.2.1. Generate Stipple Candidates

Finding the surface locations for IVS points can be completed through a texture-mapping-like approach. This process consists of two steps. First, the target surface $\mathbb{S}$ is parameterized into a UV atlas whose pixel values contain their 3D locations on the surface. Then, IVS samples lookup this atlas to find where they should be on the surface by their 2D coordinates (u_i, v_i), as is shown in Fig. 3. Those samples whose coordinates are located at vacant regions on the atlas will lose their candidacies. Fig. 2 (b) shows the distribution of the IVS points on the surface after this operation.

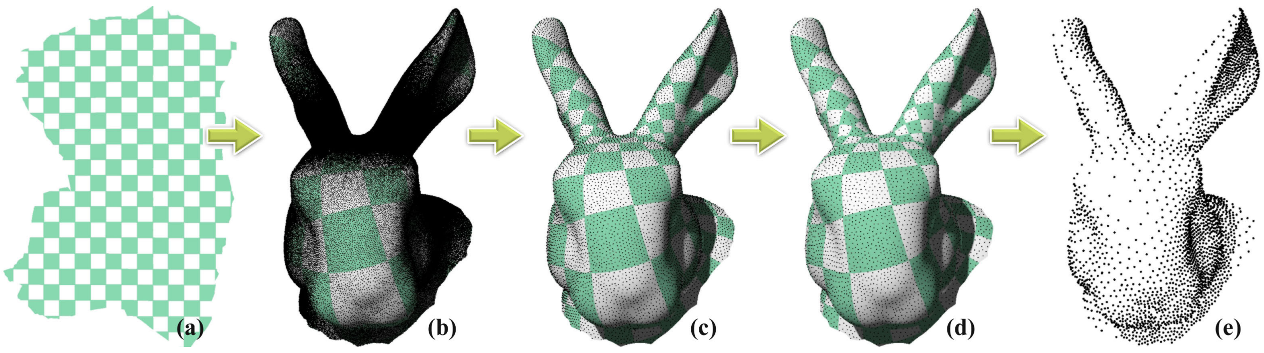

Figure 2: *Stipples generated using IVS sequences. (a) is the UV atlas with which the IVS points are anchored onto bunny's head to obtain (b). After eliminating UV distortion we obtained (c). (d) is the result when the change of areas after perspective projection are considered. Final thresholding using the shading information produces (e) which uses larger* δ.

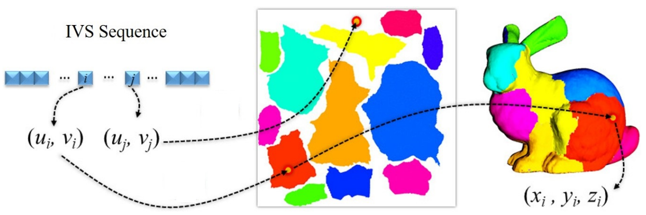

Figure 3: *Anchoring stipples. Image in the middle is an XYZ atlas which is a texture storing the object position information. Candidate stipple i with the UV coordinates* (u_i, v_i) *has a valid position* (x_i, y_i, z_i) *on the bunny. Stipple j is located at a vacant region, thus it is discarded.*

3.2.2. Eliminate Parameterizations Distortions

The unavoidable distortions to create the UV atlas caused the uneven distributions of candidates onto $\mathbb{S}$. This can be easily compensated by an area distortion factor:

$$k_{uv} = A^t_j / A^{uv}_j \tag{1}$$

where A^t_j is the area of a surface element, triangle j for example, and A^{uv}_j is the area of the same element on the atlas.

$$a^m_i \approx k_{uv} \cdot A_0 / i \begin{cases} \geq a_0 & \text{keep sample } s_i \\ < a_0 & \text{reject sample } s_i \end{cases} \tag{2}$$

where a^m_i is the surrounding area of sample i, and a_0 is the specified smallest surrounding area which is adopted to control the global sample density on $\mathbb{S}$. After this thresholding operation, the distribution of the IVS samples on $\mathbb{S}$ will appear evenly, as is shown in Fig. 2 (c).

3.2.3. Consider Area Change of Perspective Projection

The operation of Eq. 2 results in the even distribution of IVS on $\mathbb{S}$. This is for surface re-sampling instead of stippling. To satisfy criteria (1) assumed at the beginning of this section, the change of local area after perspective projection need to take into account. Suppose element j covers an area A^s_j on the projection screen thus the projection factor is $k_{proj} = A^s_j / A^t_j$, Eq. 2 becomes:

$$a^s_i \approx k_{proj} \cdot k_{uv} \cdot A_0 / i = \frac{A^s_j}{A^{uv}_j} \cdot \frac{A_0}{i} \begin{cases} \geq a_0 & \text{keep stipple } s_i \\ < a_0 & \text{reject stipple } s_i \end{cases} \tag{3}$$

where a^s_i is the surrounding area of sample i on screen. It is computed from the resolutions of the framebuffer and the size of the stipples specified by the user. By this, we obtained Fig. 2 (d) on which the stipples are evenly distributed on screen.

3.2.4. Present Shading Effect

We did not consider the shading information in the stipples in Fig. 2 (d); therefore, they are evenly distributed. Since the local tone g_i presented by stipple i on screen is $g_i = \delta_i / a^s_i$, where δ_i is the size of stipple i and a^s_i is the surrounding area of this stipple on screen, to let the stippling result *never-darker* and then the local shading result of the surface fragment g^s_i, we set:

$$g_i = \delta_i / a^s_i \begin{cases} \leq g^s_i & \text{keep stipple } s_i \\ > g^s_i & \text{reject stipple } s_i \end{cases} \tag{4}$$

The thresholding process for shading synthesis thus is as follows:

$$a^s_i \approx \frac{A^s_j}{A^{uv}_j} \cdot \frac{A_0}{i} \begin{cases} \geq \delta_i / g^s_i & \text{keep stipple } s_i \\ < \delta_i / g^s_i & \text{reject stipple } s_i \end{cases} \tag{5}$$

The final surface stippling result using this rejection strategy is shown in Fig. 2 (e). Eq. 5 covers all the aforementioned thresholding processes. In summary, the thresholding maintains the local grayness, and the goal to evenly distribute the stipples is achieved because of the intrinsic properties of IVS.

3.3. Implementation

The proposed surface stippling technique is implemented to design an efficient way to handle Eq. 5. Apparently, some of the runtime cost can be preprocessed.

3.3.1. Workflow

The proposed workflow, which is shown in Fig. 4 divides the task into two major aspects. The four modules in the gray box can be precomputed, and the four modules on the right in the green box need to be handled at runtime.

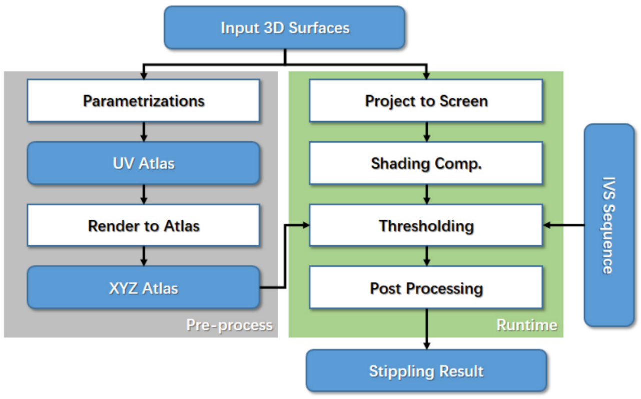

Figure 4: *Workflow of 3D surface stippling using an IVS sequence. The light blue blocks and white blocks represent data and operations, respectively.*

In the preprocessing aspect, the first module parameterizes $\mathbb{S}$ into a UV atlas M_{uv}. This module is well studied and there are various solutions. Fortunately, our proposed surface stippling technique is not sensitive to the quality of the parameterizations.

After the surface parameterizations, the geometric primitives of $\mathbb{S}$ are then rendered using the UV coordinates to generate an XYZ atlas map M_{xyz}. This map stores the object space location information of $\mathbb{S}$, such that any IVS sample s_i can obtain its location on $\mathbb{S}$ by a simple lookup using its UV coordinate (u_i, v_i). The area factor $1/A_j^{uv}$ of each fragment is also calculated and stored in M_{xyz}. The pixel value of a M_{xyz} would therefore be $(x_i, y_i, z_i, 1/A_j^{uv})$.

For the runtime operations, projecting objects onto the screen and performing shading computation are common. All results in this present paper employ a simple GPU-based rendering method with the Phong shading model, naive shadow map, and 2D texture mapping. The *Thresholding* process is for Eq. 5, and the *Post Processing* here draws the final accepted stipples using tiny disks facing the camera.

3.3.2. Parallelization

Each candidate stipple needs to go through the thresholding evaluation to determine its visibility on the final screen. Eq. 5 shows that the evaluation is independent. Therefore, by dividing the IVS sequence into several subsequences, the thresholding processes will be easily parallelized for multicore CPU/GPU implementations to generate high-quality stippling results for large images in real-time.

3.3.3. Algorithm

The complete algorithm to generate Fig. 2 (e) on is given in Alg. 1. It consists of three parts: the preprocessing, the stipple generation routine, and the postprocessing which presents the shape of the stipples.

Algorithm 1 3D Surfaces Stippling

```
1: // Part I. Pre-processing
2: Get the pre-generated 2D IVS sequence $\mathcal{S}$;
3: Divide $\mathcal{S}$ into $m$ subset $\mathcal{S}_j$, $j = 1, ..., m$;
4: for Each object $O$ in the scene do
5:     $M_{uv}$ ←Parameterize the mesh;
6:     $M_{xyz}$ ←Render to UV space;
7: end for
8: // Part II. Generating stipples for each frame
9: Specify the size of a stipple $\delta$;
10: for Each object $O$ in the scene do
11:     Render $O$ using common tech.;
12:     Calc. $A_i^s$ for each fragment in $M_{uv}$;
13:     for Each subset $\mathcal{S}_j$ do
14:         for Each sample $s_{j,i}$ in subset $\mathcal{S}_j$ do
15:             Lookup location $p_i$ of $s_{j,i}(u_i, v_i)$ on $M_{xyz}$;
16:             if $p_i$ is valid (not in vacant region) then
17:                 $(x_i, y_i, z_i, 1/A_i^{uv}) \leftarrow M_{xyz}$;
18:                 Project to screen: $(x_s, y_s, d_s) \leftarrow (x_i, y_i, z_i)$;
19:                 if $d_s$ passes depth test and is front facing then
20:                     $g_i \leftarrow$ pixel value at $(x_s, y_s)$;
21:                     $a_i \leftarrow (A_i^s / A_i^{uv}) \cdot (A_0 / i)$;
22:                     if $a_i \geq \delta / g_i$ then
23:                         Store a stipple at $(x_s, y_s)$ to $S_{stipple}$;
24:                     end if
25:                 end if
26:             end if
27:         end for
28:     end for
29: end for
30: //Part III. Post processing for each frame
31: Clear screen buffer
32: for Each stipple $s$ in $S_{stipple}$ do
33:     Draw a stipple of size $\delta$ at $(x_s, y_s)$
34: end for
```

4. Improving the Stylization

The efficiency of Alg. 1 allows users to adjust the length of the given IVS sequence and the size of a stipple δ on-the-fly for desirable results. However, using mono-black stipples of uniform size is not very expressive. Two extensions, the *varying radius* style and *multi-tone* approach are proposed to enrich the stylization without losing the real-time efficiency.

4.1. Varying Radius

The original stippling method does not restrict the stipples to maintain uniform size. Varying the stipple size can apparently capture subtle tone change better. Hence, the basic idea of varying-radius stippling is to adjust the stipple radius r such that the stipple size $\delta = \pi \cdot r^2$ matches the local tone. In this case, all accepted stipples after the evaluation of Eq. 3 would be kept and have the same surrounding area a_0 on screen. However, their sizes will be adjusted according to the local shading g_i^s on the screen. Therefore, for varying radius stippling, we use the following:

$$a_i^s \approx \frac{A_j^s}{A_j^{uv}} \cdot \frac{A_0}{i} \begin{cases} \geq a_0 & \text{keep stipple } s_i \text{ with size } \delta_i = g_i^s a_0 \\ < a_0 & \text{reject stipple } s_i \end{cases} \quad (6)$$

where a_0, as previously given, is the specified smallest surrounding area. The varying-radius method apparently adds very limited

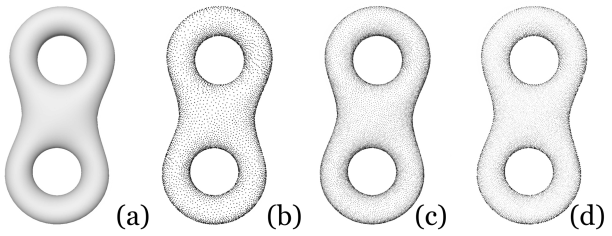

Figure 5: *Visual effect improved by the varying radius method. Image on the left (a) is the shading reference. Images in the middle (b and c) use uniform stipple size. Image on the right (d) uses the varying radius method.*

cost to the basic method; however, it generates more expressive effects, as shown in Fig. 5. Furthermore, the varying-radius method maintains better temporal coherency for dynamic scenes because stipples would change their radius before they appear or disappear in consequence frames. We employ δ_{min} in the algorithm to truncate the tiny stipples such that results would appear closer to artworks.

4.2. Multi-tone

The *multi-tone* stippling uses pens of various tones to draw the stipples. Similar to the varying-radius stippling technique, all accepted candidates after the checking of Eq. 3 would be kept except that the local brightness is lighter compared with that produced by the lightest pen. By introducing a term τ for tone, Eq. 4 becomes:

$$g_i = \tau \cdot \delta_0 / a_0$$

where δ_0 is the given stipple size. And the multi-tone stippling can be carried out as follows:

$$a_i^s \approx \frac{A_j^s}{A_j^{uv}} \cdot \frac{A_0}{i} \begin{cases} \geq a_0 & \text{keep stipple } s_i \text{ with tone } \tau = g_i^s a_0/\delta_0 \\ < a_0 & \text{reject stipple } s_i \end{cases} \tag{7}$$

Stippling artists cannot have pens of any tone. Generally, stipple tones are restricted to several selected ones. A straightforward way to tackle this is to use the closest given tone. This produces naive results with evident quantization effect, as is shown in Fig.6 **(b)**.

A better solution to eliminate the quantized effect is to mix the stipples of adjacent tones near the edges. Suppose $\tau = g_i^s a_0/\delta_0$ is the tone calculated by Eq. 7. τ^l and τ^{l+1} are two adjacent given tones that $\tau^l \leq \tau \leq \tau^{l+1}$. The selected tone can be determined as follows:

$$\tau^* = g_i^s a_i^s / \delta_0 \begin{cases} \geq \tau & \text{select } \tau^{l+1} \text{ for stipple } s_i \\ < \tau & \text{select } \tau^l \text{ for stipple } s_i \end{cases} \tag{8}$$

by which we obtain Fig.6**(c)**, on which the quantization effect vanishes. In our implementation the lightest tone τ^0 is always specified as *white*($\tau^0 = 0.0$).

5. Analysis

The proposed technique is simple, practical, and efficient. The justification for this technique, because it is for stylized rendering, is objective and not always a prerequisite. Two consensus criteria, one

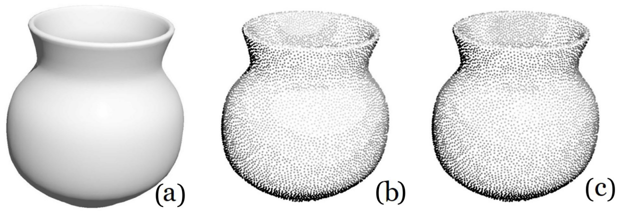

Figure 6: *Multi-tone stippling using different strategies.*

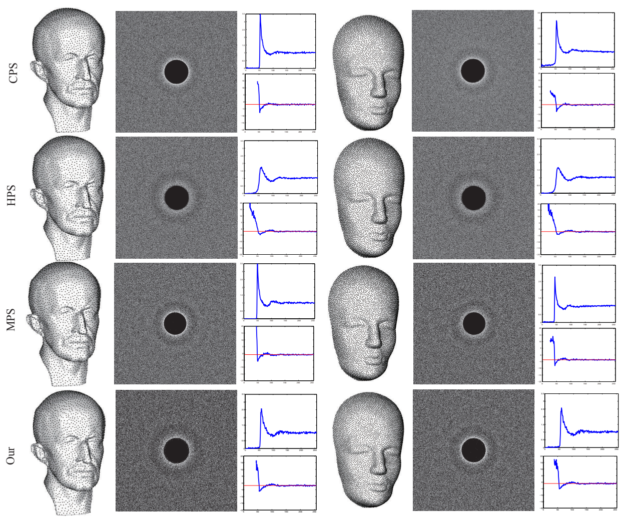

Figure 7: *Frequency analysis of different stipples distributions on surfaces. Left: stippling results; middle: power spectrum, and right: radial average (top) and anisotropy (bottom) of the power spectrum.*

on the quality of the stipple distributions and the other on the faithfulness of capturing the local tones, are adopted to evaluate our technique. Finally, for the appraisal of the proposed technique, the performance is also evaluated.

5.1. Stipples Distribution Analysis

The generally accepted standard for point distributions is the well-known blue noise properties. We expect the stipple distributions generated by our technique exhibit this property. To demonstrate, the characteristics of the generated stipples are compared with those of other blue noise sampling patterns. Note that, only the representative Poisson-disk sampling methods on surfaces, which are much faster than optimization-based approaches, are compared. The selected methods include *constrained Poisson-disk sampling* (CPS) [CCS12], *maximal Poisson-disk sampling* (MPS) [YW13], and *hierarchical Poisson-disk sampling* (HPS) [MIPS14]. *Differential domain analysis* (DDA) [WW11] is employed here to analyze the spectral properties of the point sets sampled on surfaces, including power spectrum, radially average and anisotropy. Fig. 7 shows the power spectra of point sets obtained with different methods. Although the blue noise properties of the proposed algorithm are not the best ones, they achieve relatively good spectral profiles with low anisotropy and lack of low frequencies around the origin.

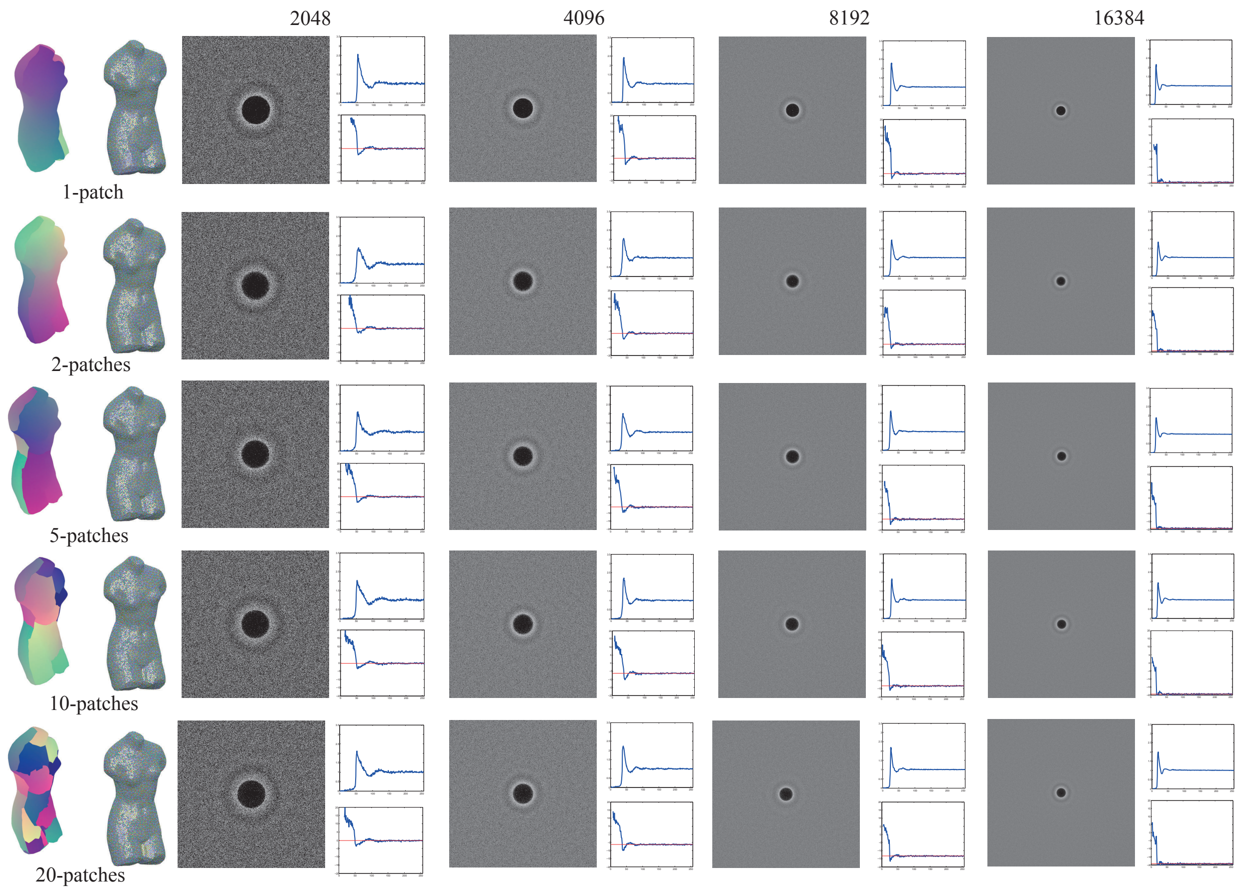

Figure 8: *Differential domain analysis of stippling results with different UV patches and incremental stippling candidates. Stipples from the incremental sets of 2048, 4096, 8192, and 16384 candidates are distinguished using red, pink, green and blue, respectively. The default settings of the DDA tool [WW11] are used to generate the spectrum profiles.*

This indicates the generated stipples are distributed randomly and uniformly manner exhibiting evident blue noise property.

We then performed the blue noise spectral analysis with different UV maps and incremental stippling candidates. Our approach is not sensitive to the number of UV patches, as is claimed before. To demonstrate this observation, the Fourier spectrum profiles generated using different numbers of UV patches are shown in Fig. 8. The blue-noise properties of stipple distributions with different numbers of stipples are also shown in Fig. 8. The smaller stipple sets are subsets of the larger ones. This example clearly illustrates that the proposed technique works well with different UV patches and incremental stippling candidates.

5.2. Expression Faithfulness

Although expressing local tone faithfully to the original realistic rendering result is not indispensable for stippling, we still want to keep this restriction, such that users would not need to re-tune the shading and other rendering parameters for the special stylization of stippling. Studying the derivation of the proposed stippling method, specifically Eqs. 5, 6, 7, and 8, we can expect that the stipples that are generated should be able to represent the tones and their variance properly.

To visually demonstrate this feature, four video sequences, each consisting of 628 frames (100 slots for one rotation round), are generated. Of these four video sequences, one is the reference realistic result. The other three are the stippling results generated by the proposed approaches under the same viewing and lighting settings.

The *peak signal-to-noise ratio* (PSNR) is a numerical measurement for tone similarity used in image quality assessment [WBSS04]. We compare the tone similarity frame by frame by computing the PSNR between each realistic and 3d stippling result pair.

The PSNR values of the three result sequences are shown in Fig. 9. Some of the selected frames of the four sequences are also presented in Fig. 10. We resized the original image (with 1024×1024 resolution) with a factor 0.4. Results of the varying-radius method (green line) matched the tones of the realistic images better compared with the results of the stippling method using uniform size (blue line). The multi-tone results (red line) even matched the realistic tones better. In addition, through numerical and visual comparisons, the proposed stippling approaches express local tone faithfully over continuous frames.

Furthermore, similar to the many image stippling methods [LM11, LM10], the *mean structural similarity measure* (SSIM) is

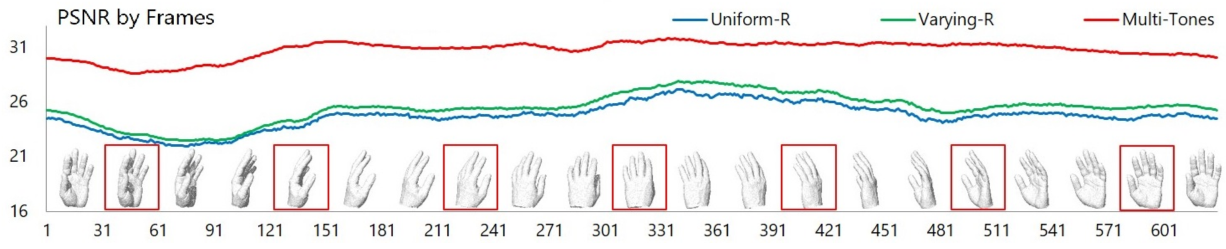

Figure 9: *Tone similarity over frames measured by PSNR. 628 continuous frames are calculated. The Y-axis starts from 16.*

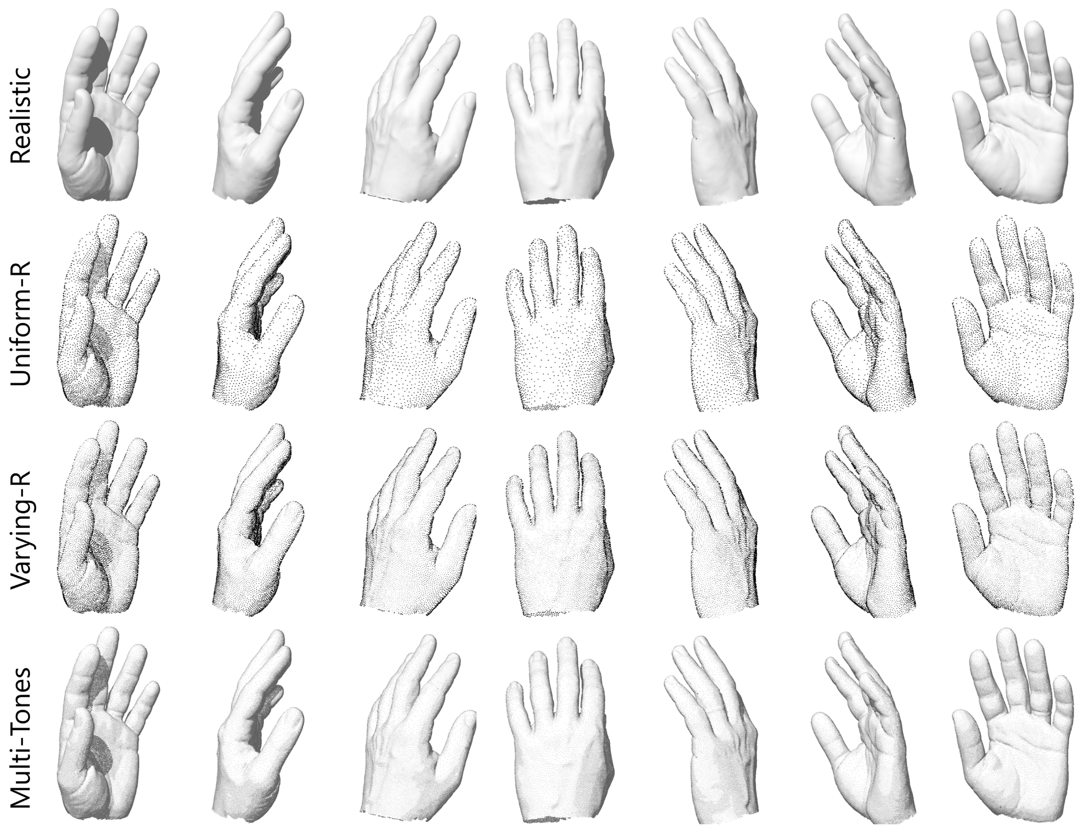

Figure 10: *Selected frames (red boxed in Fig. 9) of realistic rendering and our stipple drawings.*

used to measure the structural similarity between images. Although keeping structural information is not the objective of the proposed technique, the continuity of SSIM values over a sequence of frames can be regarded as a numerical evidence of temporal coherence for reference. The structural similarities between the three video sequences and the realistic one are shown in Fig. 11.

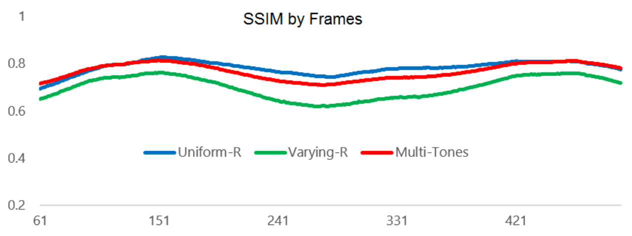

Figure 11: *Structural similarity over frames measured by SSIM. 430 continuous frames are selected. The Y-axis starts from 0.2.*

Table 1: *Performance analysis.*

Test case	Triangles	Shading(fps)	Mono(fps)	Varying Radius(fps)	Multi Tones(fps)
Bunny	41498	212	115	115	115
Kitty	21222	686	362	361	362
Eight	6144	3200	956	958	954
Hand	23186	2502	347	342	343
Max	27272	2200	290	283	282
Elephant	49918	2700	203	203	203
Scenes1	133841	2400	96	93	91
Scenes2	292640	1600	62	61	62

5.3. Performance

The proposed methods are implemented on GPU for most parallelizable phases such as shading, post rendering, stylization improvement and thresholding. We use a lazy mode that the rele-

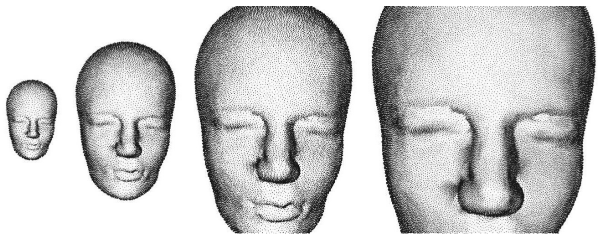

Figure 12: *Stippling a zoom-in sequence.*

vant information for calculating a stipple is only to be updated when necessary. For example, the projection area of a mesh on the screen is only updated when the camera is changed or any animations are applied. Therefore, the performance is fluctuating at runtime. The *lowest* GPU-based performance data for each input scene are captured and shown in Table 1. All test cases and algorithms are conducted on a PC with a 4.0GHz Intel® Core^{TM} i7-6700k CPU, 32GB RAM, and a Nvidia® Geforce® GTX 1080 Graphics Card. Based on Table 1, we can infer that the proposed approaches achieved real-time frame rate.

6. Experimental results

The proposed technique is applied on various 3D objects and scenes to verify the feasibility, robustness, and visual quality of the generated stippling results.

6.1. Temporal Coherence

Our method enforced temporal consistency. Each stipple candidate has its unique location on the mesh surface and will move with the surface in dynamic scenes. The thresholding stipples generated by IVS ensured that only a very small number of stipples will appear or vanish frame by frame.

Three test cases are selected to test the temporal coherence of the stippling results. The first one, as shown in Fig. 12, is a zoom-in sequence, which is generated by pushing the camera closer to the model. Our technique generates stipples incrementally while maintaining a constant apparent tone. The second test is to stipple a static surface with dynamic lighting directions and a moving camera. Some frames of this test are shown in Fig. 10. The video of this test shows that the stipples are incrementally changed, and only a few stipples appear/disappear between sequential frames. The final coherence test is to stipple an animation sequence, as shown in Fig. 14.

We refer readers to the accompanying video for the complete animations of the three tests. From the figures and video, we can see that each individual frame appears close to a stippling artwork. In addition, the flicker between successive frames is negligible in the cases of moving cameras, changing lightings, or deforming meshes.

6.2. Extended Stippling Styles

In contrast to many other surface stippling methods, our technique can produce locally varying stipple radii or present multiple tone level stipples. Fig. 15 illustrates stippling results with varying stipple sizes and multi-tones. These stippling results greatly improved the visual quality compared with the stippling of uniform size with mono-black color. These extensions provide users with more options for expressions..

6.3. Stippling Complex Scenes

A good robustness test is to stipple complex scenes, as shown in Fig.13. This scene contains 21 objects (surfaces) which are composed of 292,640 triangular facets. We show the stippling results with varying point sizes. The frame rate is around 150 fps on the GTX 1080 graphics card for a 2048×1600 output under dynamic viewing and lighting conditions. This example demonstrates that the proposed technique is able to synthesize high-quality stippling results in real-time for large scenes.

6.4. Parameterizations Independence

The generated stipple distributions with different numbers of UV patches that exhibit the blue noise property are statistically presented in Sec. 5.1. To further verify the robustness and the validity we employed different surface parameterization methods, including *least squares conformal maps* (LSCM) [LPRM02], *Iso-chart texture atlasing* (UVAtlas) [ZSGS04], and *boundary first flattening* (BFF) [SC17], to generate the UV maps for our stippling, as shown in Fig. 1. They are visually similar. This test illustrates again that our algorithm is not restricted to the choice of parameterizations method.

6.5. Comparison with Other Incremental Sequences

For 2D stippling, many decent algorithms provide good qualities of progressive blue noise in real-time, such as [Uli93], and tile-based methods [KCODL06, ANHD17]. These methods were compared with IVS in terms of quality, speed and usability in [MCQS18]. In our study, we compare the IVS with two types of other point sequences, a random sequence and two of the low-discrepancy sequences (e.g., Halton [Hal60], Sobol [Sob67] and its variant Scrambled Sobol). These sequences are chosen because they can be easily integrated into the proposed 3D stippling framework and maintain parallelizable attributes. The visual quality of stipples on surfaces using the IVS sequence, a random sequence, and Sobol and Halton sequences are compared in Fig. 16. As observed, surface stippling using the IVS sequence shows evident visual superiorities.

7. Conclusion and Future Works

We presented novel techniques to generate stipples for dynamic 3D objects/scenes in real-time. The key idea is to map a 2D incremental and parallel IVS sequence to the surfaces by carefully eliminating the distortions caused by parameterizations and the change of geometric primitives' areas after perspective projections. The shading effect of the scenes is also considered for the stipple generation. To further enrich the visual effect, we propose two extensions to generate stippling with varying-radius and multi-tone styles, the tone similarity among continuous frames are preserved. Our approach achieves real-time frame rate and the distribution of the generated stipples exhibits good blue-noise properties. The incremental property of the IVS sequence results in good temporal coherence between frames.

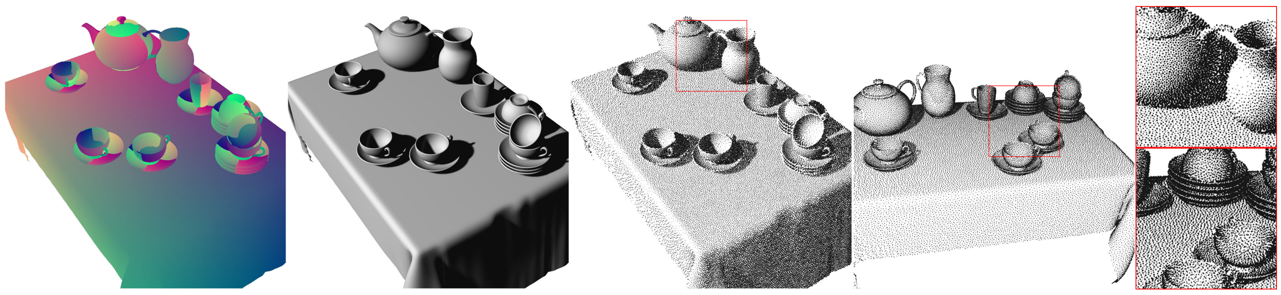

Figure 13: *Instant stippling for a complex 3D scene. From left to right: the UV coordinates, the realistic rendering result, two stippling results with varying point sizes from different views, and the zoom-in details.*

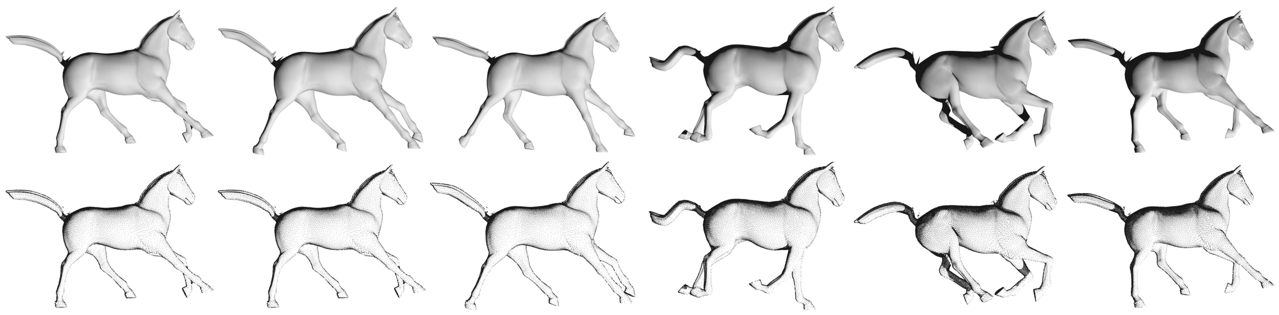

Figure 14: *Instant stippling for an animation sequence from [SP04]. The frame numbers from left to right are 1, 2, 3, 42, 46, and 48. The lighting is continuously changing during the whole animation process.*

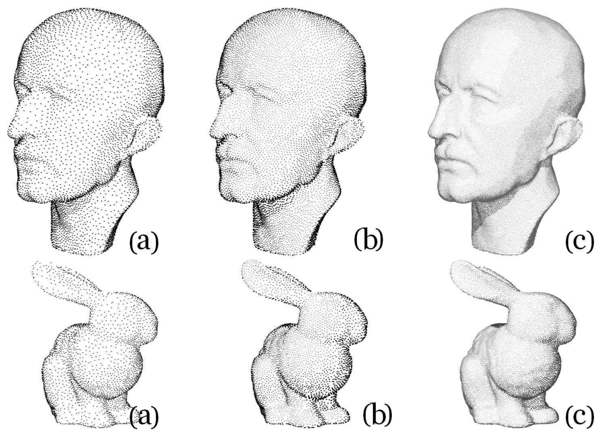

Figure 15: *Comparison of different stippling styles. From left to right are stippling results using uniform size with mono-black color stipples, varying radius stipples, and multi-tones stipples.*

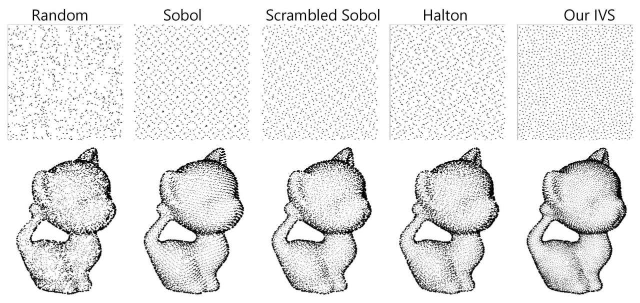

Figure 16: *Comparison with other incremental sample sequences. The 2D point sequences are shown on the top and the related stippling results on the bottom.*

The proposed method has the following limitations. First, the final results could be affected by the parameterizations approaches. Serious distortions on UV space may introduce decline on the final stipple distributions, as is shown on the last two results in Fig. 1. Second, the proposed technique is not good at capturing high-frequent shading effects like thin sharp hard shadows. Third, the proposed technique cannot handle fuzzy or transparent surfaces, nor can it deal with volumetric objects.

In the future, we would like to further reduce UV distortions by considering deformation tensor. Furthermore, adding points dynamically rather than removing is a potential option for improving the performance. The technique proposed by Ahmed et al. [ANHD17] is another alternative for the proposed stippling framework because it supports parallelization. Moreover, using IVS sequence to synthesize other NPR styles, such as hatching and painting, is also interesting.

Acknowledgments. We thank anonymous reviewers for their valuable comments and suggestions. The research of Jianwei Guo and Dongming Yan is partially funded by the Beijing Natural Science Foundation (4184102), the National Natural Science Foundation of China (61772523,61802406,61620106003). The research of Hanqiu Sun was supported by the National Natural Science Foundation of China (61602183) and the UGC direct grant (4055060).

References

[AGY*17] Ahmed A., Guo J., Yan D.-M., Franceschi J.-Y., Zhang X., Deussen O.: A simple push-pull algorithm for blue-noise sampling. *IEEE Trans. on Vis. and Comp. Graphics 23*, 12 (2017), 2496–2508. 2

[AHD15] Ahmed A. G. M., Huang H., Deussen O.: AA patterns for point sets with controlled spectral properties. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 34*, 6 (2015), 212:1–212:8. 2

[ANHD17] Ahmed A. G. M., Niese T., Huang H., Deussen O.: An adaptive point sampler on a regular lattice. *ACM Trans. on Graphics (Proc. SIGGRAPH) 36*, 4 (2017), 138:1–138:13. URL: http://doi.acm.org/10.1145/3072959.3073588, doi:10.1145/3072959.3073588. 2, 9, 10

[APC*16] Ahmed A., Perrier H., Coeurjolly D., Ostromoukhov V., Guo J., Yan D., Huang H., Deussen O.: Low-discrepancy blue noise sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 35*, 6 (nov 2016), 247:1–247:13. 2

[Bay73] Bayer B. E.: An optimum method for two-level rendition of continuous-tone pictures. *IEEE Int. Conf. on Communications 26* (1973), 11–15. 2

[Bri07] Bridson R.: Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH Sketches* (2007), p. 22. doi:10.1145/1278780.1278807. 2

[BSD09] Balzer M., Schl T., Deussen O.: Capacity-constrained point distributions : A variant of lloyd ' s method. *ACM Trans. on Graphics (Proc. SIGGRAPH)* (2009). 2

[BTBP07] Baer A., Tietjen C., Bade R., Preim B.: Hardware-accelerated Stippling of Surfaces derived from Medical Volume Data. In *EurographicsIEEE-VGTC Symposium on Visualization* (2007), Museth K., Moeller T., Ynnerman A., (Eds.), The Eurographics Association. 2

[BWWM10] Bowers J., Wang R., Wei L.-Y., Maletz D.: Parallel Poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 29*, 6 (2010), 166:1–166:10. doi:10.1145/1882261.1866188. 2

[CCS12] Corsini M., Cignoni P., Scopigno R.: Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. on Vis. and Comp. Graphics 18*, 6 (2012), 914–924. 2, 6

[CJW*09] Cline D., Jeschke S., White K., Razdan A., Wonka P.: Dart throwing on surfaces. *Computer Graphics Forum (Proc. EGSR) 28*, 4 (2009), 1217–1226. 2

[Coo86] Cook R. L.: Stochastic sampling in computer graphics. *ACM Trans. on Graphics 5*, 1 (1986), 69–78. 2

[CYC*12] Chen Z., Yuan Z., Choi Y. K., Liu L., Wang W.: Variational blue noise sampling. *IEEE Trans. on Vis. and Comp. Graphics 18*, 10 (2012), 1784. 2

[DFG99] Du Q., Faber V., Gunzburger M.: Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review 41* (1999), 637–676. 2

[dGBOD12] de Goes F., Breeden K., Ostromoukhov V., Desbrun M.: Blue noise through optimal transport. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), 171. 2

[DHvOS00] Deussen O., Hiller S., van Overveld C. W. A. M., Strothotte T.: Floating points: A method for computing stipple drawings. *Computer Graphics Forum 19*, 3 (2000), 41–50. doi:10.1111/1467-8659.00396. 2

[DI13] Deussen O., Isenberg T.: Halftoning and stippling. In *Image and Video-Based Artistic Stylisation*. Springer London, 2013, pp. 45–61. doi:10.1007/978-1-4471-4519-6_3. 2

[DSZ17] Deussen O., Spicker M., Zheng Q.: Weighted linde-buzo-gray stippling. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 36*, 6 (2017), 233:1–233:12. doi:10.1145/3130800.3130819. 2

[DW85] Dippé M. A. Z., Wold E. H.: Antialiasing through stochastic sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 19*, 3 (July 1985), 69–78. URL: http://doi.acm.org/10.1145/325165.325182, doi:10.1145/325165.325182. 2

[EMP*12] Ebeida M. S., Mitchell S. A., Patney A., Davidson A. A., Owens J. D.: A simple algorithm for maximal poisson-disk sampling in high dimensions. *Computer Graphics Forum (Proc. EUROGRAPHICS) 31*, 2 (2012), 785–794. 2

[EPM*11] Ebeida M. S., Patney A., Mitchell S. A., Davidson A., Knupp P. M., Owens J. D.: Efficient maximal Poisson-disk sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30*, 4 (2011), 49:1–49:12. 2

[Fat11] Fattal R.: Blue-noise point sampling using kernel density model. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30*, 4 (2011), 48:1–48:12. doi:10.1145/2010324.1964943. 2

[GM09] Gamito M. N., Maddock S. C.: Accurate multidimensional Poisson-disk sampling. *ACM Trans. on Graphics 29*, 1 (2009), 8:1–8:19. 2

[GYJZ15] Guo J., Yan D.-M., Jia X., Zhang X.: Efficient maximal Poisson-disk sampling and remeshing on surfaces. *Computers & Graphics 46*, 6-8 (2015), 72–79. 2

[Hal60] Halton J. H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik 2*, 1 (1960), 84–90. 9

[Jon06] Jones T. R.: Efficient generation of Poisson-disk sampling patterns. *Journal of Graphics Tools 11*, 2 (2006), 27–36. 2

[KCODL06] Kopf J., Cohen-Or D., Deussen O., Lischinski D.: Recursive wang tiles for real-time blue noise. *ACM Trans. on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 509–518. 2, 9

[KW07] KrÃijger J., Westermann R.: Efficient stipple rendering. In *IADIS Computer Graphics and Visualization* (Lisbon, 2007), pp. 19–26. 3

[LD08] Lagae A., Dutre P.: A comparison of methods for generating poisson disk distributions. *Computer Graphics Forum 27*, 1 (2008), 114–129. 2

[LM10] Li H., Mould D.: Contrast-aware halftoning. *Computer Graphics Forum 29*, 2 (May 2010), 273–280. doi:10.1111/j.1467-8659.2009.01596.x. 7

[LM11] Li H., Mould D.: Structure-preserving stippling by priority-based error diffusion. In *Proceedings of Graphics Interface 2011* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2011), GI '11, Canadian Human-Computer Communications Society, pp. 127–134. URL: http://dl.acm.org/citation.cfm?id=1992917.1992938. 7

[LMT*03] Lu A., Morris C. J., Taylor J., Ebert D. S., Hansen C., Rheingans P., Hartner M.: Illustrative interactive stipple rendering. *IEEE Trans. on Vis. and Comp. Graphics 9*, 2 (2003), 127–138. doi:10.1109/TVCG.2003.1196001. 1

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (July 2002), 362–371. URL: http://doi.acm.org/10.1145/566654.566590, doi:10.1145/566654.566590. 1, 9

[LVPI18] Lawonn K., Viola I., Preim B., Isenberg T.: A survey of surface-based illustrative rendering for visualization. *Computer Graphics Forum 37* (2018). 3

[MARI17] Martín D., Arroyo G., Rodríguez A., Isenberg T.: A survey of digital stippling. *Computers & Graphics 67* (2017), 24 – 44. doi:https://doi.org/10.1016/j.cag.2017.05.001. 1, 2

[MCQS18] Ma L., Chen Y., Qian Y., Sun H.: Incremental voronoi sets for instant stippling. *The Visual Computer 34*, 6 (Jun 2018), 863–873. URL: https://doi.org/10.1007/

s00371-018-1541-7, doi:10.1007/s00371-018-1541-7. 2, 3, 9

[MEA*18] MITCHELL S. A., EBEIDA M. S., AWAD M. A., PARK C., PATNEY A., RUSHDI A. A., SWILER L. P., MANOCHA D., WEI L.-Y.: Spoke-darts for high-dimensional blue-noise sampling. *ACM Trans. on Graphics 37*, 2 (2018), 22:1–22:20. 2

[MIPS14] MEDEIROS E., INGRID L., PESCO S., SILVA C.: Fast adaptive blue noise on polygonal surfaces. *Graphical Models 76*, 1 (2014), 17–29. 2, 6

[MP92] MITSA T., PARKER K. J.: Digital halftoning technique using a blue-noise mask. *J. Opt. Soc. Am. A 9*, 11 (Nov. 1992), 1920–1929. 2

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast hierarchical importance sampling with blue noise properties. *ACM Trans. on Graphics (Proc. SIGGRAPH) 23*, 3 (2004), 488–495. 2, 3

[Ost07] OSTROMOUKHOV V.: Sampling with polyominoes. *ACM Trans. Graph. 26*, 3 (July 2007), 78. URL: http://doi.acm.org/10.1145/1276377.1276475, doi:10.1145/1276377.1276475. 2

[PFS03] PASTOR O. M., FREUDENBERG B., STROTHOTTE T.: Real-time animated stippling. *IEEE Computer Graphics & Applications 23*, 4 (2003), 62–68. 1, 3

[PHWF01] PRAUN E., HOPPE H., WEBB M., FINKELSTEIN A.: Real-time hatching. *Proceedings of Siggraph* (2001), 581. 3

[PS04] PASTOR O. M., STROTTHOTE T.: Graph-based point relaxation for 3d stippling. In *Mexican International Conference in Computer Science* (2004), pp. 141–150. 3

[PZ07] PALACIOS J., ZHANG E.: Rotational symmetry field design on surfaces. *ACM Trans. on Graphics (Proc. SIGGRAPH) 26*, 3 (2007), 55:1–55:10. 3

[QGY*16] QUAN W., GUO J., YAN D.-M., MENG W., ZHANG X.: Analyzing surface sampling patterns using the localized pair correlation function. *Computational Visual Media 2*, 3 (2016), 219–230. 2

[SBB16] SUAREZ J., BELHADJ F., BOYER V.: Real-time 3d rendering with hatching. *The Visual Computer* (2016), 1–16. 3

[SC17] SAWHNEY R., CRANE K.: Boundary first flattening. *ACM Trans. Graph. 37*, 1 (Dec. 2017), 5:1–5:14. doi:10.1145/3132705. 1, 9

[Sec02] SECORD A.: Weighted Voronoi stippling. *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR '02 1* (2002), 37. doi:10.1145/508535.508537. 2

[SFWS03] SOUSA M. C., FOSTER K., WYVILL B., SAMAVATI F.: Precise Ink Drawing of 3D Models . *Computer Graphics Forum 22*, 3 (2003), 369–379. 3

[Sob67] SOBOL' I. M.: On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki 7*, 4 (1967), 784–802. 9

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. on Graphics (Proc. SIGGRAPH) 23*, 3 (2004), 399–405. 10

[Uli87] ULICHNEY R.: *Dithering with Blue Noise*. MIT Press, 1987. 2

[Uli93] ULICHNEY R. A.: Void-and-cluster method for dither array generation. In *Human Vision, Visual Processing, and Digital Display IV* (sep 1993), Allebach J. P., Rogowitz B. E., (Eds.), vol. 1913, SPIE. URL: https://doi.org/10.1117/12.152707, doi:10.1117/12.152707. 2, 9

[VBTS07] VANDERHAEGHE D., BARLA P., THOLLOT J., SILLION F. X.: Dynamic Point Distribution for Stroke-based Rendering. In *Rendering Techniques* (2007), Kautz J., Pattanaik S., (Eds.), The Eurographics Association. 1, 2

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612. doi:10.1109/TIP.2003.819861. 7

[WCE07] WHITE K. B., CLINE D., EGBERT P. K.: Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing* (2007), pp. 129–132. 2

[Wei08] WEI L.-Y.: Parallel poisson disk sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 27*, 3 (2008), 20:1–20:9. doi:10.1145/1360612.1360619. 2

[Wei10] WEI L.-Y.: Multi-class blue noise sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 4 (2010), 79:1–79:8. doi:10.1145/1778765.1778816. 2

[WLY*16] WANG X., LE T. H., YING X., SUN Q., HE Y.: User controllable anisotropic shape distribution on 3d meshes. *Computational Visual Media 2*, 4 (2016), 305–319. 3

[WPC*14] WACHTEL F., PILLEBOUE A., COEURJOLLY D., BREEDEN K., SINGH G., CATHELIN G., DE GOES F., DESBRUN M., OSTROMOUKHOV V.: Fast tile-based adaptive sampling with user-specified fourier spectra. *ACM Trans. on Graphics (Proc. SIGGRAPH) 33*, 4 (2014), 56:1–56:11. 2

[WW11] WEI L.-Y., WANG R.: Differential domain analysis for non-uniform sampling. *ACM Transactions on Graphics (TOG) 30*, 4 (2011), 50. 6, 7

[XLGG11] XU Y., LIU L., GOTSMAN C., GORTLER S. J.: Computers & Graphics Capacity-Constrained Delaunay Triangulation for point distributions. *Computers and Graphics 35*, 3 (2011), 510–516. 2

[YGW*15] YAN D.-M., GUO J., WANG B., ZHANG X., WONKA P.: A survey of blue-noise sampling and its applications. *J. Comput. Sci. Technol 30*, 3 (2015), 439–452. 2

[YNZC05] YUAN X., NGUYEN M. X., ZHANG N., CHEN B.: Stippling and Silhouettes Rendering in Geometry-Image Space. In *Eurographics Symposium on Rendering (2005)* (2005), Bala K., Dutre P., (Eds.), The Eurographics Association. 1, 3

[Yuk15] YUKSEL C.: Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum (Proc. EUROGRAPHICS) 34*, 2 (2015), 25–32. 2

[YW13] YAN D.-M., WONKA P.: Gap processing for adaptive maximal Poisson-disk sampling. *ACM Trans. on Graphics 32*, 5 (2013), 148:1–148:15. doi:10.1145/2516971.2516973. 2, 6

[ZSGS04] ZHOU K., SYNDER J., GUO B., SHUM H.-Y.: Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), ACM, pp. 45–54. 1, 9