PaintersView: Automatic Suggestion of Optimal Viewpoints for 3D Texture Painting

Yuka Takahashi
The University of Tokyo
yukatkh@is.s.u-tokyo.ac.jp

Tsukasa Fukusato
The University of Tokyo
tsukasafukusato@is.s.u-tokyo.ac.jp

Takeo Igarashi
The University of Tokyo
takeo@acm.org

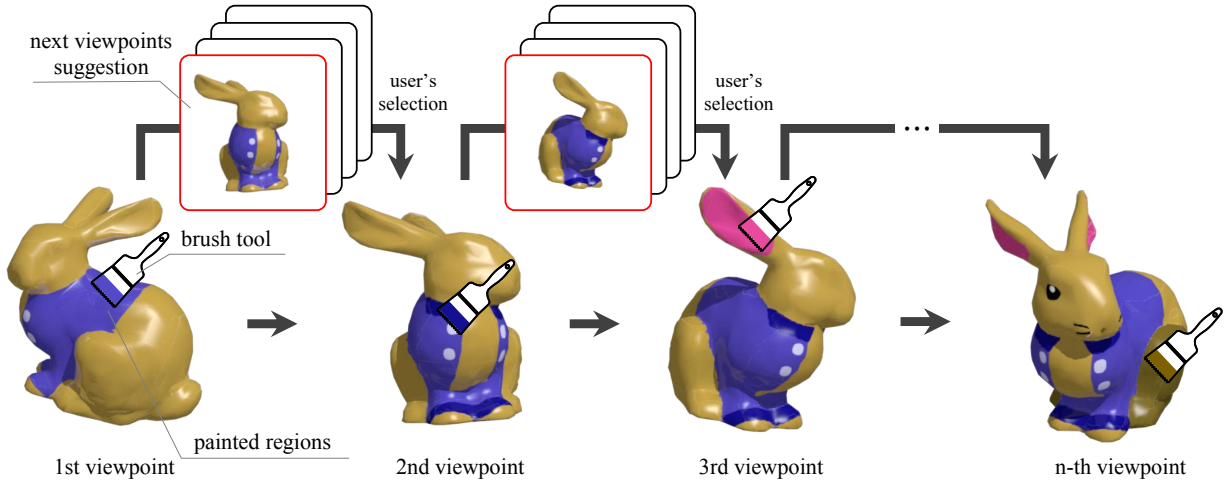

Figure 1: Concept of Our System. We envision that software equips a "next-best-view" button to sequentially optimize a next viewpoint that is capable of displaying the unpainted regions as painting progresses on a model.

ABSTRACT

Although 3D texture painting has an advantage of making it easy to grasp the overall shape compared with a method of drawing directly onto a UV map, a disadvantage is unpainted (or distorted) regions appearing in the result due to, for example, self-occluded parts. Thus, in order to perform painting without leaving unpainted parts, sequential change of viewpoints is necessary. However, this process is highly time-consuming. To address this problem, we propose an automatic suggestion of optimal viewpoints for 3D texture painting. As the user paints a model, the system searches for optimal viewpoints for subsequent painting and presents them as multiple suggestions. The user switches to a suggested viewpoint by clicking on a suggestion. We conducted a user study and confirmed that the proposed workflow was effective for 3D texture painting envisioned by users.

SA '19 Technical Briefs, November 17–20, 2019, Brisbane, QLD, Australia

ACM ISBN 978-1-4503-6945-9/19/11…$15.00
https://doi.org/10.1145/3355088.3365159

CCS CONCEPTS

• **Computing methodologies** → Graphics system and interface.

KEYWORDS

3D texture painting, viewpoint selection, self-occlusion

ACM Reference Format:
Yuka Takahashi, Tsukasa Fukusato, and Takeo Igarashi. 2019. PaintersView: Automatic Suggestion of Optimal Viewpoints for 3D Texture Painting. In *SIGGRAPH Asia 2019 Technical Briefs (SA '19 Technical Briefs), November 17–20, 2019, Brisbane, QLD, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3355088.3365159

1 INTRODUCTION

3D texture painting system is intuitive and convenient for directly designing hand-painted textures on 3D models when compared with 2D space painting (in which the users paint on a UV map). The painting process follows these three steps; (i) specifying viewpoints to make the unpainted areas visible from the front side, (ii) drawing the brush strokes on the screen, and then (iii) re-projecting the painted strokes in the 3D views. However, it can be cumbersome to control viewpoints manually and requires time-consuming work. In addition, when painting the self-occluded regions, the user must repeatedly perform viewpoint operation to make unpainted regions visible for further painting. Here, our goal is to find possible "next" viewpoints where the unpainted areas are easily visible to users for a given model (see Fig 1).

Our core observation is that 3D painting is a sequential task (the situation changes every moment), so intermediate result of painting (painted texture and annotations) are important for viewpoint selection, as well as the shape features of the 3D model. Thus, the proposed method considers two characteristics; (i) geometry information, and (ii) intermediate paint results at painting time. In addition, in order to reduce the number of viewpoint control operations, we provide two interactive mechanisms ; (i) an annotation tool to limit the viewpoints candidates, and (ii) making front-most surface translucent to allow direct painting of occluded regions. Our primary contributions are summarized as follows:

- A novel concept to interactively find optimal viewpoints specialized for the 3D painting task.
- A novel algorithm to estimate optimal viewpoints by using geometry and information of painted regions.

2 RELATED WORK

3D Texture Painting Support: LayerPaint [Fu et al. 2010] allows users to directly display self-occluded faces and draw on them. Also, several systems to locally control viewpoints around 3D models have been proposed [Khan et al. 2005; Ortega and Vincent 2014]. While these techniques are useful for some situations, especially when drawing of long curves on 3D surfaces, the user must manually find "good" viewpoints which are capable of displaying unpainted areas. Our method only selects viewpoints that serves as a concrete starting point for users, so the method does not interfere with also applying the previous methods.

Human-Powered Viewpoint Support: Applying human perception allows users to optimize preferable parameter sets, such as the positions and the orientation of the viewpoint. For example, Koyama et al. [2014] propose a crowdsourcing method to analyze high-dimensional parameter spaces in order to obtain a distribution of human preference. In addition, Chen et al. [2014] propose a history-based approach to suggest viewpoints around 3D models. However, these algorithms cannot obtain the parameter analysis results in real-time because they use either the crowdsourcing platform or the recording data, so it remains difficult to use them for a 3D texture painting task. Therefore, we simply utilize the painting results (current state), and sequentially suggest "good" viewpoints which is suitable for real-time use.

Next Best View System: In the fields of computer vision and robotics research, "next best view" estimation is used for efficient 3D model creation from images [Dunn and Frahm 2009; Massios et al. 1998; Pito 1999]. The system estimates new view-points that reduces the reconstructed model uncertainty. Hence, we utilize these concepts and propose a method to find viewpoints specialized for the 3D painting task.

3 SYSTEM OVERVIEW

In this system, a user performs 3D texture painting alongside with viewpoint selection. As with traditional frameworks for 3D texture painting, the user can select particular painting tools (e.g., brush tools, and color selection), and directly draw on the 3D surface, as shown in Figure 2. The four next viewpoints suggestions are updated automatically after each drawing operation, and the user chooses the next viewpoint from those candidates. Based on the new viewpoints, the user repeats this process until they get satisfied, thereby obtaining the desired 3D painting results.

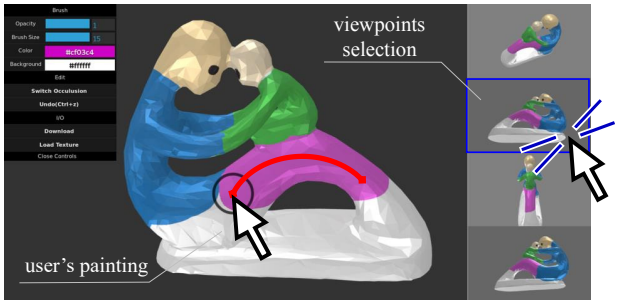

Figure 2: User Interface. The user directly paints on the 3D surface from the current viewpoint (left). After painting, the system enables the users to select the next viewpoint (right).

4 METHOD

4.1 Viewpoints Estimation

Viewpoints are uniformly sampled around the input model $\vec{v}_i$, $i \in \{1, \cdots, N\}$, and the system searchers for optimal viewpoints for subsequent painting (in this paper, we empirically set $N = 70$). Note that the direction of each viewpoint $\vec{d}_i$ is a direction vector from the location of the viewpoint camera to the origin of the current model. Given a current painted result, the optimization problem is defined as follows:

$$\underset{i \in \{1, \cdots, N\}}{\arg\max} \; E_{geometry} + w\, E_{paint} \tag{1}$$

where w is a weight value for tweaking the balance between $E_{geometry}$ and E_{paint}. In this paper, we empirically found $w = 0.04$ provided a good balance.

To verify whether a face f is visible in a given i-th viewpoint, we check that (1) the projection of face f falls in the viewpoint screen range, and (2) the normal is directed toward the viewpoints. Then, based on the angle information between the face normal and the view direction, we define a normal-based function to maximize the number of the visible faces.

$$E_{geometry} = \frac{1}{|F_i|} \sum_{f \in F_i} \|\vec{d}_i \cdot \vec{n}_f\| \tag{2}$$

where F_i is a set of faces that are visible from the i-th viewpoint (i.e., the front-most visible surface) and $\vec{n}_f$ is a face normal.

In order to reflect the current paint result, we compute the face area A_f projected onto the i-th viewpoint. Then, we also define a simple heuristic that prefers viewpoints which are capable of displaying the unpainted area, namely

$$E_{paint} = \frac{1}{|F_i^p|} \sum_{f \in F_i^p} A_f \tag{3}$$

where F_i^p is a set of unpainted faces in F_i.

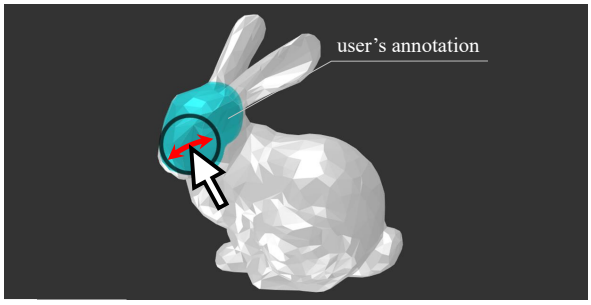

Figure 3: Annotation Tool. When the user specifies regions (cyan), the system then updates the next viewpoints so that the user-specified regions are more clearly visible.

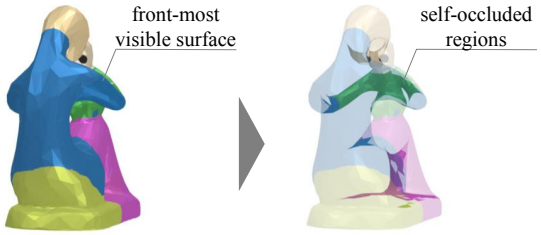

Figure 4: Occlusion Access Tool. The front-most visible surface on the screen becomes transparent and the user can directly paint previously occluded regions.

4.2 User Interaction

Annotation Tool: One problem is that the viewpoints are automatically computed by Eq. (1) (without any constraints) and the user interaction is not considered in this equation, which may cause problems when drawing on 3D models with semantic meanings (e.g., the facial features of human-like characters). One straightforward approach is to segment the input model into semantic regions in advance. However, this requires the careful tuning of multiple parameters, and it is difficult to consider the intermediate paint results on the surface model.

Therefore, we implement an annotation tool to allow the simple specification of semantic regions by hand, as shown in Figure 3. When the user performs a mouse-drag operation with the brush tool, the system selects all the faces F_a lying along the mouse trajectory. The system then finds the next viewpoints that are capable of displaying the annotated faces based on $E_{annotation} = 1/|F_a| \sum_{f \in F_a} \|\vec{d}_i \cdot \vec{n}_f\|$.

Occlusion Access Tool: To address problems that drawing self-occluded areas is challenging with manual painting, we implement a function to access self-occluded faces without changing viewpoints that was inspired by LayerPaint [Fu et al. 2010] (see Figure 4). The front-most visible surface on the screen becomes translucent and the user can directly paint the previously occluded regions (which are now visible).

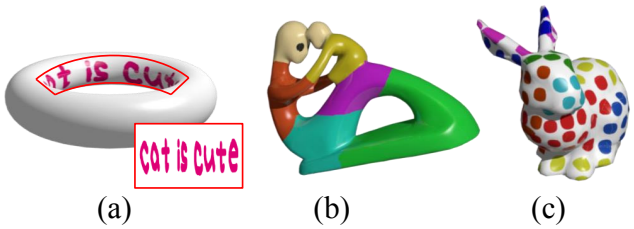

Figure 5: The rendered and textured models: (a) torus, (b) fertility & (c) the Stanford Bunny. Each participant was given their unpainted versions and was asked to re-create them with painting tools.

Viewpoint Selection: In order to expand the variety of suggested viewpoints, our system automatically chooses four different viewpoint candidates for each step by using the proposed score (see Eq.(1)), and the user can interactively select the next viewpoint from the proposed options. In this paper, we sort the scores of each viewpoint by descending order into a vector. We first chooses the top-ranked candidate, and then greedily choose three additional candidates with minimum overlap of visible faces. The selected four viewpoints are shown on the right-hand side in Figure 2.

5 IMPLEMENTATION

Our prototype system was implemented on a web browser using JavaScript. The system can export the textured models in a common file format, so that users can use them in existing software such as Autodesk Maya and Blender.

6 USER STUDY

We conducted two user studies; (1) one is to compare our method with the fully manual setting of viewpoints, and (2) the other is to assess the effectiveness of the real-time painting features when compared with only the normal-based function (see Eq.(2)). We invited six participants, aged 20 to 30 years old, to perform the tasks. All participants were casual users who had at least one year's experience with 3D or painting software. For each task, half of the participants were asked to perform a baseline method first and our system second, and the other half were required to perform the tasks in the opposite order.

Task 1. Manual Control vs Our Method: We provided the participants with two unpainted models (a torus model and a fertility model) and asked them to attempt to reproduce the look of the textured models until they were satisfied (see Fig 5). Participants were asked to replicate one sentence on the self-occluded area of the torus model first, and then to color the fertility model according to a reference figure. After completing these tasks, all participants were asked to answer two questions: (Q1) Which did you find was more comfortable to use for design? (manual or our method) and (Q2) Score the overall usability of each method (1-5, with 1 = "extremely dissatisfied," and 5 = "extremely satisfied").

Task 2. w/o paint feature vs w/ paint feature: We provided participants the Stanford bunny model and asked them to uniformly draw a polka dot pattern (see Fig. 5(c)) using our interface (with

Table 1: Comparison (Manual vs Our Method).

Model	#		Manual	Our Method
Torus	Q1	Vote [%]	16.7 (=1/6)	83.3 (=5/6)
	Q2	Mean	2.75	4.50
		SD	0.76	0.84
Fertility	Q1	Vote [%]	16.7 (=1/6)	83.3 (=5/6)
	Q2	Mean	2.58	3.92
		SD	1.11	0.66

Table 2: Comparison (w/o paint feature vs w/ paint feature).

#		w/o Paint Feature	w/ Paint Feature
Q1	Vote [%]	33.3(=2/6)	66.7(=4/6)
Q2	Mean	3.17	4.00
	SD	0.75	0.89

intermediate paint results) and again with the normal-based function Eq.(2) (i.e., without consideration of intermediate paint results). At the end of the creation process, participants filled out a questionnaire consisting of two questions: (Q1) Which did you find was more comfortable to use for design ? (normal-based function or our method) and (Q2) Score the overall usability of each method (1-5, 1 = "extremely dissatisfied," and 5 = "extremely satisfied").

Tables 1 and 2 shows the post-experimental questionnaire results, giving the mean values and the standard deviations. In general, our method was rated more positively on both models by the participants, and the participants were satisfied with it (> 3). The intermediate paint results also produced higher ratings than the normal-based function. The results show that our system has the potential to interactively provide more plausible viewpoints in a way that will benefit users. The participants' comments regarding our system are summarized below:

- *P1: After selecting viewpoints estimated with normal-based method (i.e., w/o the intermediate paint results), I often fine-tuned the parameters of the next viewpoint in Task 2. But, with the real-time updating, I thought that it was easier to interactively find best viewpoints without slight adjustments.*
- *P2: In the case of simple models such as a sphere, full-manual control is enough to design 3D textured models. However, the viewpoint candidates are very useful for understanding the shape of the 3D models and current textured results at painting time.*

However, some participants had difficulty painting 3D surface, as described below.

- *P3: I thought that it was difficult to plan where to draw (e.g., eye regions, followed nose) in advance, so I wanted to directly customize the viewpoint candidates, for example, the opposite side of the current viewpoints.*

In addition, Figure 6 shows the average number of "manual" control operations and the number of viewpoint selection operations to design the textured models in each task. This confirmed that our approach can significantly reduce the number of repetitive manual operations.

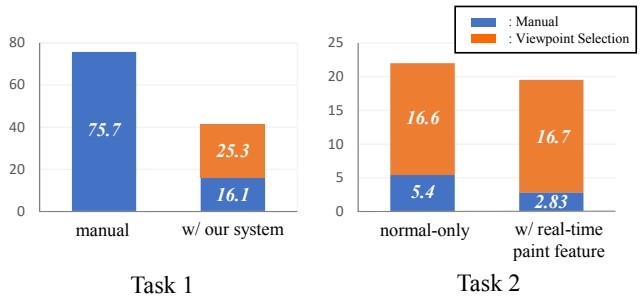

Figure 6: The average number of "manual" control and "viewpoint selection" operations in the user study.

7 LIMITATION AND FUTURE WORK

Our current system only supports the design of a single-layer texture on a 3D surface (however, the base color is excluded). It might be better to allow users to design multi-layer texture painting if necessary. We also plan to develop a function to automatically refine the wrongly-painted regions when the painted pixel color is different from neighbors by using the mesh's connectivity.

8 CONCLUSION

In this paper, we have proposed an interactive system to find the next viewpoints for drawing on the unpainted areas. To support texture painting activities, we utilized the shape features and the intermediate paint results at painting time. The proposed system was rated higher on average by the users who had experience in using 3D modeling software.

ACKNOWLEDGEMENT

This work was supported by JSPS KAKENHI grant No. JP17H00752 & JP19K20316, Japan.

REFERENCES

Hsiang-Ting Chen, Tovi Grossman, Li-Yi Wei, Ryan M Schmidt, Björn Hartmann, George Fitzmaurice, and Maneesh Agrawala. 2014. History assisted view authoring for 3D models. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2027–2036. https://doi.org/10.1145/2556288.2557009

Enrique Dunn and Jan-Michael Frahm. 2009. Next Best View Planning for Active Model Improvement. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 53:1–53:11. https://doi.org/10.5244/C.23.53

Chi-Wing Fu, Jiazhi Xia, and Ying He. 2010. LayerPaint: A Multi-Layer Interactive 3D Painting Interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 811–820. https://doi.org/10.1145/1753326.1753445

Azam Khan, Ben Komalo, Jos Stam, George Fitzmaurice, and Gordon Kurtenbach. 2005. Hovercam: Interactive 3D Navigation for Proximal Object Inspection. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*. ACM, 73–80. https://doi.org/10.1145/1053427.1053439

Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-Powered Parameter Analysis for Visual Design Exploration. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 65–74. https://doi.org/10.1145/2642918.2647386

Nikolaos A Massios, Robert B Fisher, et al. 1998. A best next view selection algorithm incorporating a quality criterion. In *Proceedings of the 9th British Machine Vision Conference*. Department of Artificial Intelligence, University of Edinburgh, 78:1–78:10. https://doi.org/10.5244/C.12.78

Michaël Ortega and Thomas Vincent. 2014. Direct Drawing on 3D Shapes with Automated Camera Control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2047–2050. https://doi.org/10.1145/2556288.2557242

Richard Pito. 1999. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 10 (1999), 1016–1030. https://doi.org/10.1109/34.799908