Supplementary materials: Microstructure-Aware Woven Fabric Generation

Yingjie Tang[1†] Di Luo[1†] Zixiong Wang[1] Xiaoli Ling[2] Jian Yang[1,2] Beibei Wang[2*]
[1]Nankai University [2]Nanjing University

Table 1. Parameters predicted by WeavingLLM. The top four parameters affect weaving draft and procedural geometry, and the rest are shading parameters.

Parameter	Type/Range	Description
D	binary matrix	weaving pattern draft
c_{ply}	int $\{1, 3, 4, 6\}$	number of plies in yarn
k_{sliding}	float (0, 1)	yarn sliding strength
k_{flyaway}	float (0, 1)	flyaway fiber amount
T	float (0, 1)	thickness of weft and warp
α_{weft}	float (0, 1)	roughness of wefts
α_{warp}	float (0, 1)	roughness of warps

Table 2. Full set of yarn parameters.

property	range	description
c_{ply}	$\{1, 3, 4, 6\}$	number of plies
w	$(0, 1)$	width of yarn
h_f	$(0, 2)$	height field scaling factor
$u_{\max}$	$(0, 90)$	maximum inclination angle
ψ	$(0, 90)$	fiber twist angle
α	$(0, 90)$	ply twist angle

1. Implementation details

1.1. Training details

Macro-scale texture generator We fine-tune the FLUX diffusion model using the SimpleTuner framework, a lightweight configuration-driven toolkit for parameter-efficient adaptation of Hugging Face Diffusers models. SimpleTuner enables LoRA and LyCORIS fine-tuning through minimal modification of JSON configuration files.

We collect a dataset of 600 microstructure-free fabric textures along with the corresponding description. The dataset covers a variety of representative patterns, including but not limited to stripe, plaid, camouflage, and fancy prints, as shown in Fig. 1.

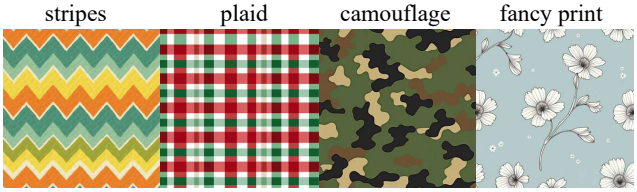


Figure 1. Examples from our fabric texture dataset. All images are expected to be free from yarn-level microstructure.

We adopt the standard LoRA scheme, where trainable rank-decomposed matrices are injected into the U-Net denoiser's attention modules. We set the LoRA rank to 64 and apply a learning rate of 1e-4, optimizing the model using the AdamW optimizer with polynomial learning rate scheduling and a warm-up of 100 steps. We train for a total of 30,000 steps on 600 microstructure-free fabric textures, costing 24 hours on a single NVIDIA RTX 4090 GPU.

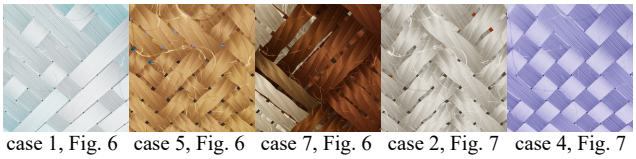
WeavingLLM We fine-tune a pre-trained large language model, Qwen2.5-14B-it [10], with QLoRA [2]. The model weights were quantized to 4-bit NF4 precision, and the LoRA adapters are injected into all linear modules. We set the LoRA rank to 32 and apply a learning rate of 2e-4, optimizing the model using the AdamW optimizer with polynomial learning rate scheduling and a warm-up of 170 steps. We apply instruction tuning on a dataset of 1,142 annotated weaving drafts (which is represented as a binary matrix). The instruction is shown in Appendix 2. We set the batch size to 4 and train for a total of 5,680 steps, costing 5 hours on a single NVIDIA RTX 4090 GPU.

1.2. Rendering details

Our framework generates macro-scale texture and procedural microstructure separately, and employs fusion rendering using a fabric shading model. We choose the SpongeCake model [12], a layered shading model for rendering, as previous works [4, 9, 14] have demonstrated its effectiveness for capturing fabric specular reflections.

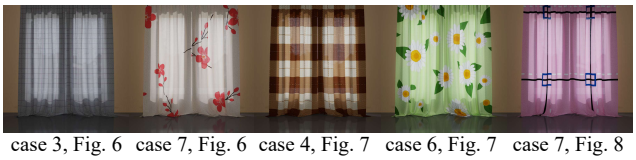
We employ a three-layer configuration within the

* Corresponding author.
† These authors contribute equally.
[1] College of Computer Science, Nankai University, Tianjin, China
[2] School of Intelligence Science and Technology, Nanjing University, Suzhou, China



case 1, Fig. 6 case 5, Fig. 6 case 7, Fig. 6 case 2, Fig. 7 case 4, Fig. 7

Figure 2. Close-up renderings of several cases.



case 3, Fig. 6 case 7, Fig. 6 case 4, Fig. 7 case 6, Fig. 7 case 7, Fig. 8

Figure 3. Transmission renderings of several cases.

SpongeCake [12] model. The top layer accounts for flyaway fibers, while the second and third layers represent weft and warp yarns, respectively, with their vertical ordering determined by the weaving pattern. The roughness and thickness of the weft and warp layers are the shading parameters given by WeavingLLM. We also supports transmission rendering, as Fig. 3 shows.

Similar to Jin et al. [4], we add an additional Lambertian term to the SpongeCake model to approximate multiple scattering. We further apply a height-based ambient occlusion (AO) factor to account for yarn-level shadowing and masking. Our full bidirectional scattering distribution function (BSDF) is given by:

$$f(\omega_i, \omega_o) = f_s(\omega_i, \omega_o) + \frac{h - h_{\min}}{h_{\max} - h_{\min}} \frac{k_d}{2\pi}, \quad (1)$$

where f_s is the specular reflectance term given by the SpongeCake model, h is the local height, $h_{\max}$ and $h_{\min}$ are the maximum and minimum yarn heights, and k_d denotes the albedo. This term effectively approximates inter-yarn shadowing and masking at a low computational cost.

To incorporate fiber-level details, we apply precomputed 1D normal and visibility maps on the ply cross-section, following the approach of Montazeri et al. [6]. These 1D maps represent the distribution of fibers across the cross-sectional profile of the ply. We refer readers to their original work for more details. This technique enables realistic close-up rendering, as displayed in Fig. 2.

1.3. Flyaway fibers

Flyaway refers to fibers escaping from the fabric surface. To model this, we propose to introduce an additional fiber layer with stochastic fiber orientations in the SpongeCake model, where the 3D fiber orientation $\mathbf{o_{flyaway}}$ is constructed from two 2D Perlin noise N_1, N_2, where N_1 controls the fiber position and horizontal orientation, and N_2 controls the vertical orientation.

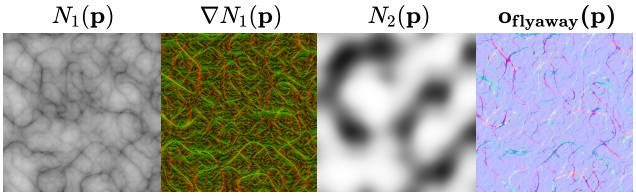


Figure 4. The stochastic fiber orientation $\mathbf{o_{flyaway}}$ is constructed from two noises: N_1 determines the positions of fibers and its gradient reveals the horizontal orientation, and N_2 determines vertical orientation.

Specifically, we first construct N_1 by introducing a sharpened Perlin noise and adding up 4 levels of frequencies through fractal Brownian motion (fBm). We treat the zero isosurface of each level as flyaway fibers. Leveraging the analytical gradient of the Perlin noise, we derive the 2D horizontal orientation $\mathbf{o_{2d}}$:

$$N_1(\mathbf{p}) = \sum_{i=0}^{3} 2^{-i} \sqrt{|P(2^i \mathbf{p})|}, \quad (2)$$

$$\nabla N_1(\mathbf{p}) = \sum_{i=0}^{3} \frac{\pm \nabla P(2^i \mathbf{p})}{2\sqrt{|P(2^i \mathbf{p})|}}, \quad (3)$$

$$\mathbf{o_{2d}} = \text{normalize}(-\nabla N_1(\mathbf{p}).y, \nabla N_1(\mathbf{p}).x), \quad (4)$$

where $\mathbf{p} = (x, y)$ denotes the 2D position on the UV plane, P denotes standard 2D Perlin noise and ∇P denotes its gradient. Note that flipping ∇N_1 does not affect the orientation calculation, we ignore the sign for simplicity.

Then, we use an external uniform Perlin noise N_2 to control the vertical orientation. Combining the two noise results in 3D fiber orientation $\mathbf{o_{flyaway}}$.

$$N_2(\mathbf{p}) = \text{CDF}_\text{P}(P(\mathbf{p})), \quad (5)$$

$$\mathbf{o_{flyaway}}(\mathbf{p}) = \text{normalize}(\mathbf{o_{2d}}, N_2(\mathbf{p})), \quad (6)$$

where the cumulative distribution function of Perlin noise CDF_P is used to construct uniform distribution.

To reproduce the highlight behavior of flyaway fibers, we set the roughness of the flyaway layer to 0.1. To enhance the visibility of fiber regions (isosurface of N_1), we set the layer thickness to:

$$T = k_{\text{flyaway}} \min(0.1, 4\min(1.0, 0.2||\nabla N_1(\mathbf{p})||)^4), \quad (7)$$

as we found the length of gradient $||\nabla N_1||$ smoothly capture the fiber area from N_1. When $T < 0.1$, we set $\mathbf{o_{flyaway}} = (0, 0, 1)$ to introduce base sheen effect. The thickness is scaled by k_{flyaway} to enable user control. For intuition, we demonstrate the distributions of N_1, ∇N_1, N_2, and the resulting $\mathbf{o_{flyaway}}$ in Fig. 4.

1.4. Details of curved helix model

We propose a procedural curved helix yarn model to map ply-level geometry directly from UV coordinates, depicted

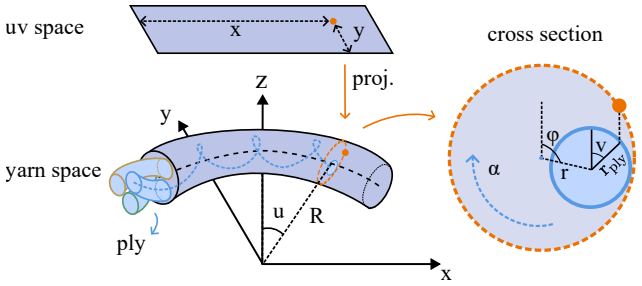


Figure 5. Curved helix model for multi-ply yarns. The model achieves analytical normal/orientation formulation, where the yarn space arc parameters u, v are linear mapped from UV space.

in Fig. 5. We model yarn centerline as a circular arc, while individual plies form helices around the yarn centerline with rotation speed α. We parameterize the plies through yarn space arc coordinates u and v, which are linearly projected from surface UV coordinates x, y. Therefore, the rotational phase $\varphi(u)$ of plies along the axis is:

$$\varphi(u) = \varphi(0) + uR\alpha, \tag{8}$$

where R is the radius of the yarn arc, $\varphi(0)$ is the initial phase of the ply, where each ply has a different initial phase.

Without loss of generality, we derive the formulations based on the floating segment of weft yarns. The ply centerline C is a curve parameterized by u:

$$C(u) = \begin{bmatrix} x_0 + R\sin(u) \\ -r\sin(\varphi(u)) \\ z_0 + R\cos(u) + r\cos(\varphi(u)) \end{bmatrix}, \tag{9}$$

where x_0 and z_0 are constants that do not affect the results, r is the distance from the ply center to the yarn center, and u_{max} is the maximum inclination angle of the yarn arc.

The orientation of the ply $\mathbf{o}_{\text{ply}}$ is the derivative of $C(u)$ with respect to u:

$$\mathbf{o}_{\text{ply}}(u) = \frac{\partial C(u)}{\partial u} = R\begin{bmatrix} \cos(u) \\ -r\alpha\cos(\varphi(u)) \\ -\sin(u) - r\alpha\sin(\varphi(u)) \end{bmatrix}. \tag{10}$$

The fiber orientation $\mathbf{t}$ is obtained by rotating the ply orientation around the normal at a certain twist angle ψ:

$$\mathbf{t}(u,v) = \text{rotate}(\mathbf{o}_{\text{ply}}, \mathbf{n}, \psi) \tag{11}$$

note that the orientation should be normalized for final use.

The normal $\mathbf{n}$ is the vector from ply center to ply surface point u, v:

$$\mathbf{n}(u,v) = \begin{bmatrix} \sin(u)\cos(v) \\ \sin(v) \\ \cos(u)\cos(v) \end{bmatrix}. \tag{12}$$

And the height h contains the height of yarn centerline, and the offset of ply centerline:

$$h(u,v) = R\cos(u) + h_{\text{ply}}(u,v), \tag{13}$$
$$h_{\text{ply}}(u,v) = r\cos(\varphi(u)) + r_{\text{ply}}\cos(u)\cos(v). \tag{14}$$

In addition, we apply a height field scaling mechanism with a factor h_f, which affects both $\mathbf{n}, \mathbf{t}$ and h in the final stage. We refer readers to Jin et al. [4]'s paper for further details.

For each yarn segment $x \in [x_{\min}, x_{\max}], y \in [0, 1]$, we map a surface point from UV space x, y to yarn space u, v by applying linear projection for each ply:

$$u = 2u_{\max}\frac{x - x_{\min}}{x_{\max} - x_{\min}} - u_{\max}, \tag{15}$$
$$v = \frac{-y - (\frac{sin(\varphi(u)}{2} - r_{\text{ply}})}{sin(\varphi(u) + 2r_{\text{ply}}}\pi - \frac{\pi}{2}. \tag{16}$$

where $u_{\max}$ is the maximum inclination angle of yarn segment, and w is the width yarn.

We list all the parameters for configuring a yarn in Tab. 2. In our automatically generated results, c_{ply} is predicted by WeavingLLM, and we set w to 0.95, α to 60. For single-ply yarns, we set h_f to 0.2, $u_{\max}$ to 35, and ψ to 0. For multi-ply yarns, we set h_f to 0.7, $u_{\max}$ to 80, and ψ to 10. These empirical values cover most cases. For simplicity, we do not make every parameter flexible, although they can be expanded as needed.

1.5. Instructions for WeavingLLM

We provide two key instructions required by WeavingLLM: a set of fabric expertise and priors for prompt tuning the model to predict fabric parameters (Appendix 1), and the instruction for weaving draft generation (Appendix 2). The full set of parameters predicted by WeavingLLM is summarized in Tab. 1.

1.6. User study

In Fig. 6, we provide two examples in the user study of fabric material generation. In Fig. 7, we provide two examples in the user study of weaving draft generation.

2. More results

Text conditioning We provide additional fabric material generation results under text conditioning in Fig. 8, 9 and 10, and compare them with DressCode [3] and MatFuse [11].

Image conditioning We provide additional fabric material generation results under both text and image conditioning, and compare them with MatFuse [11] in Fig. 11.

Figure 6. Two examples in the user study of fabric material generation. Given text prompt and three anonymous renderings, the users are requested to select the result that matches the prompt and appears most realistic. The order of the examples is *MatFuse, Ours, DressCode* (left), and *Ours, DressCode, MatFuse* (right).

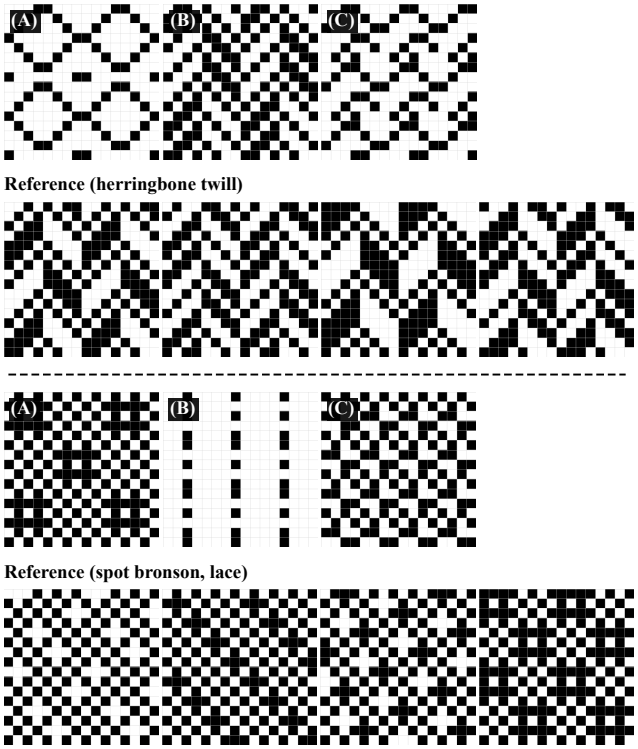


Figure 7. Two examples in the user study of weaving draft generation. Given text prompt and three anonymous generated weaving drafts, the users are requested to select the result that best matches the prompt and given references. The order of the examples is *GPT-5, WeavingLLM, Qwen2.5* (top), and *WeavingLLM, Qwen2.5, GPT-5* (bottom).

We include an extra comparison with MaterialPicker [5] in Fig. 12. We thank the authors for providing their data, which enables a direct comparison with the results reported in their paper. The comparison shows that both methods can generate textures that closely match the input image. However, benefiting from the procedural microstructures, our method produces significantly richer details in close-up renderings, requiring only an additional caption describing the weaving pattern.

Weaving draft generation We provide additional weaving draft generation results under text conditioning in Fig. 13 and 14. Further comparative results are shown in Fig. 15, where we compare WeavingLLM with Qwen2.5 [10] and GPT-5 [7]. For each case, we present more reference drafts and compute three metrics: the average cosine similarity of the Fourier spectra between the generated draft and all reference drafts (COSSIM), user study preference rates (PREF), and the average learned perceptual image patch similarity (LPIPS) [13] between the renderings of generated and reference drafts.

References

[1] Kris Bruland. Handweaving.net, 2025. https://handweaving.net/. 13

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. 1

[3] Kai He, Kaixin Yao, Qixuan Zhang, Jingyi Yu, Lingjie Liu, and Lan Xu. Dresscode: Autoregressively sewing and generating garments from text guidance. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024. 3, 6, 7, 8

[4] Wenhua Jin, Beibei Wang, Miloš Hašan, Yu Guo, Steve Marschner, and Ling-Qi Yan. Woven fabric capture from a single photo. In *Proceedings of SIGGRAPH Asia 2022*, 2022. 1, 2, 3

[5] Xiaohe Ma, Valentin Deschaintre, Miloš Hašan, Fujun Luan, Kun Zhou, Hongzhi Wu, and Yiwei Hu. Materialpicker: Multi-modal dit-based material generation. *ACM Transactions on Graphics (TOG)*, 44(4):1–12, 2025. 4, 10

[6] Zahra Montazeri, Søren B. Gammelmark, Shuang Zhao, and Henrik Wann Jensen. A practical ply-based appearance model of woven fabrics. *ACM Trans. Graph.*, 39(6), 2020. 2

[7] OpenAI. Introducing GPT-5. https://openai.com/index/introducing-gpt-5/, 2025. Accessed: YYYY-MM-DD. 4, 13

[8] Sam Sartor and Pieter Peers. MatFusion: A Generative Diffusion Model for SVBRDF Capture. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023. 9

[9] Yingjie Tang, Zixuan Li, Milos Hasan, Jian Yang, and Beibei Wang. Woven fabric capture with a reflection-transmission photo pair. In *ACM SIGGRAPH 2024 Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 1

[10] Qwen Team. Qwen2.5: A party of foundation models, 2024. 1, 4, 13

[11] Giuseppe Vecchio, Renato Sortino, Simone Palazzo, and Concetto Spampinato. MatFuse: Controllable Material Generation with Diffusion Models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4429–4438, 2024. 3, 6, 7, 8, 9

[12] Beibei Wang, Wenhua Jin, Miloš Hašan, and Ling-Qi Yan. Spongecake: A layered microflake surface appearance model. *ACM Transactions on Graphics*, 42(1), 2022. 1, 2

[13] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 4

[14] Junqiu Zhu, Christophe Hery, Lukas Bode, Carlos Aliaga, Adrian Jarabo, Ling-Qi Yan, and Matt Jen-Yuan Chiang. A realistic multi-scale surface-based cloth appearance model. In *ACM SIGGRAPH 2024 Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 1

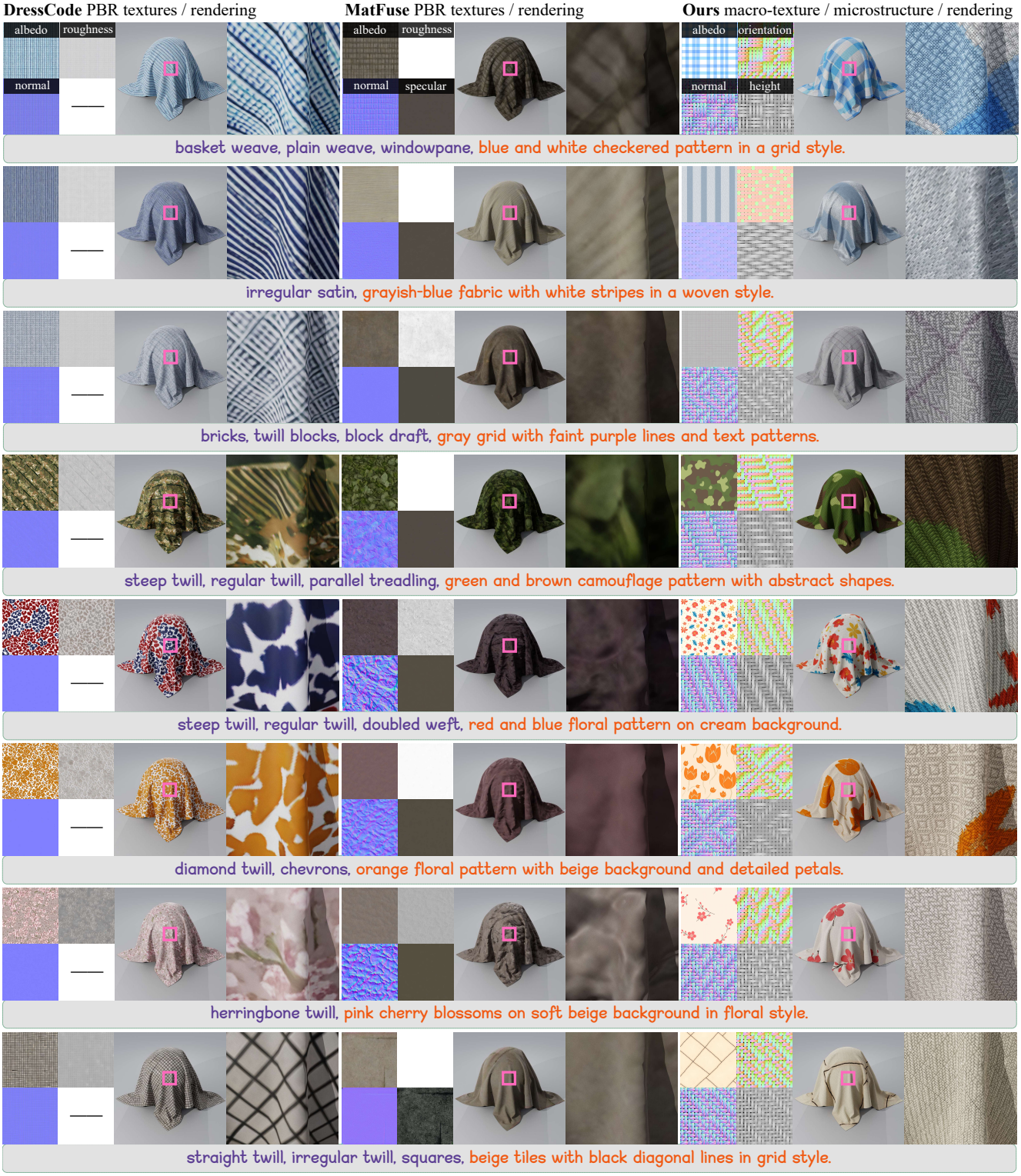


Figure 8. **Comparison on text conditioning.** Given text prompt, DressCode [3] and MatFuse [11] generate PBR textures, which struggle to capture fine fabric details, resulting in blurring and distortion in close-up views. In contrast, our method generates procedural microstructure together with macroscale texture, ensuring a realistic appearance in both macro and close-up views.

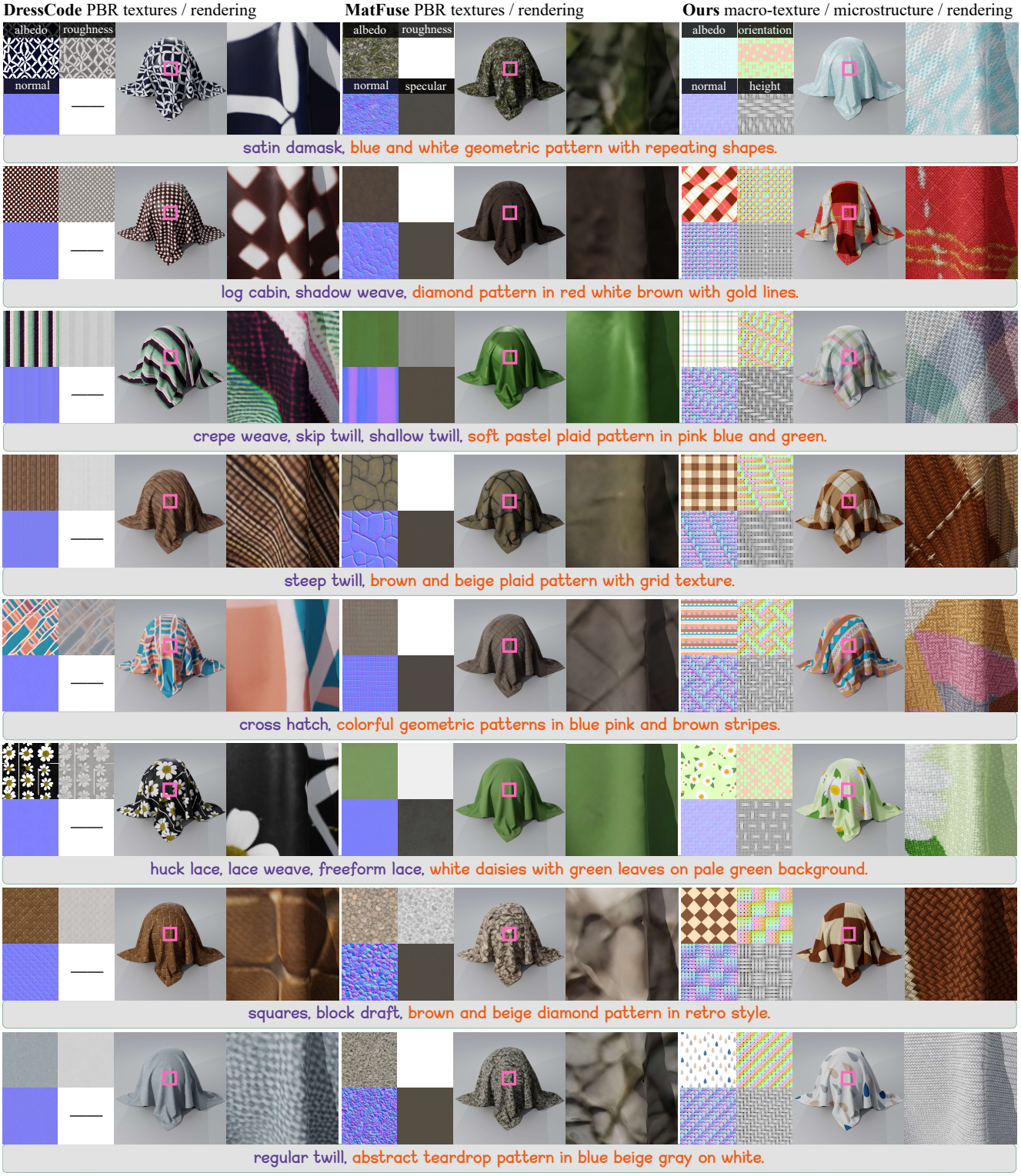


Figure 9. **Comparison on text conditioning.** Given text prompt, DressCode [3] and MatFuse [11] generate PBR textures, which struggle to capture fine fabric details, resulting in blurring and distortion in close-up views. In contrast, our method generates procedural microstructure together with macroscale texture, ensuring a realistic appearance in both macro and close-up views.

Figure 10. **Comparison on text conditioning.** Given text prompt, DressCode [3] and MatFuse [11] generate PBR textures, which struggle to capture fine fabric details, resulting in blurring and distortion in close-up views. In contrast, our method generates procedural microstructure together with macroscale texture, ensuring a realistic appearance in both macro and close-up views.

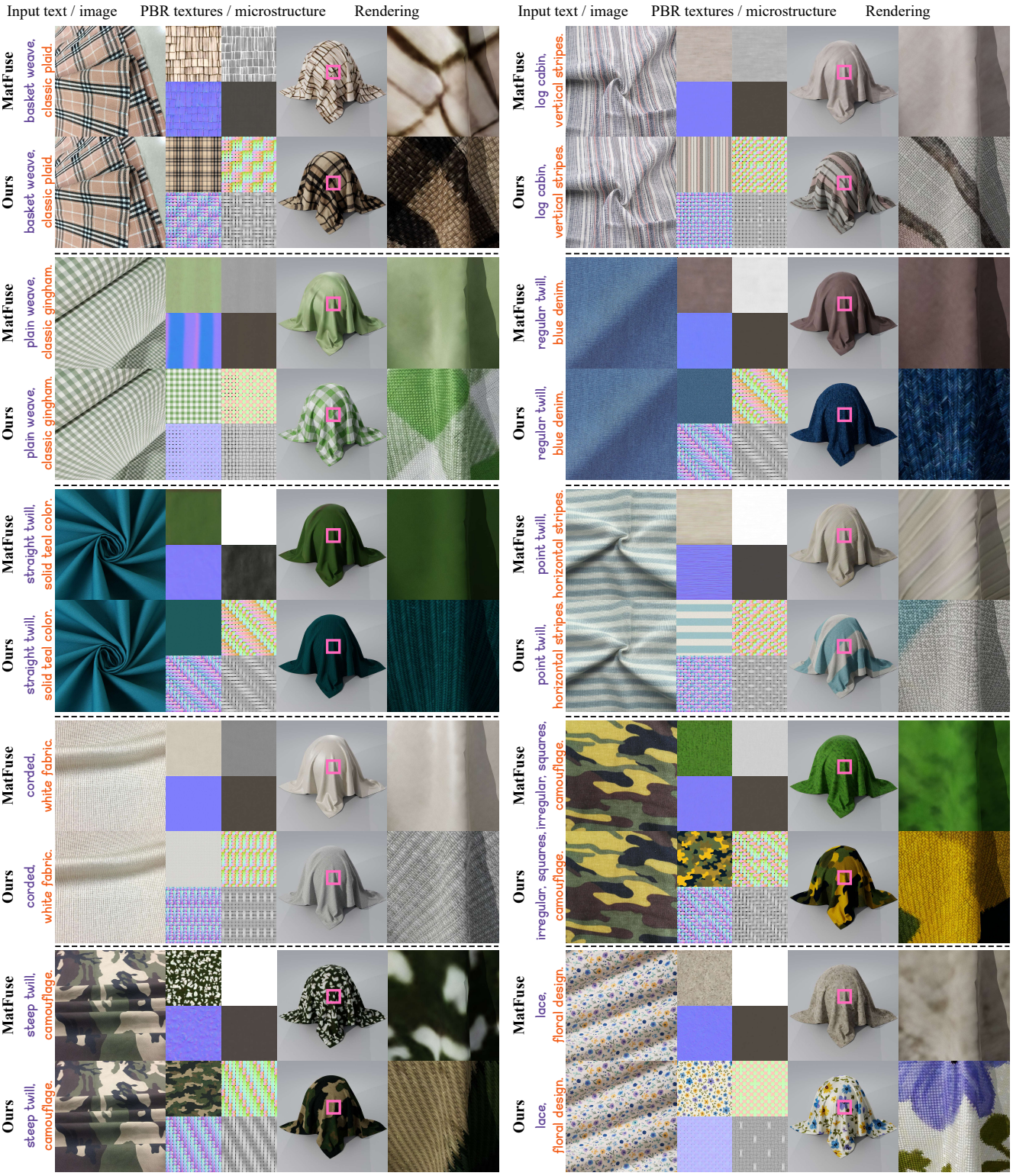


Figure 11. **Comparison with MatFuse [11] on text and image conditioning**. Given both text and image prompts, our method generates textures that closely match the input image, achieving better consistency and richer detail than MatFuse [8].

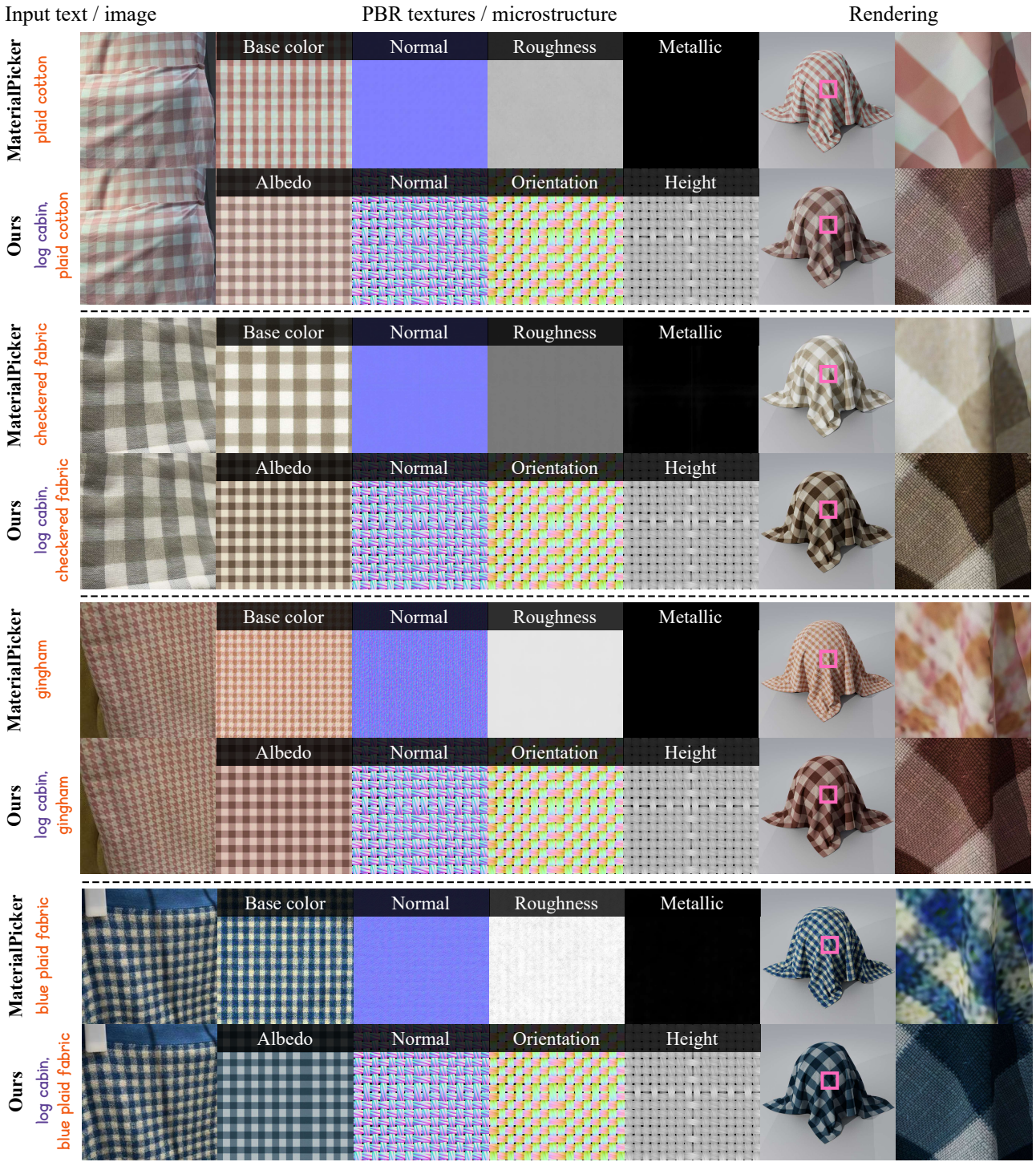


Figure 12. **Comparison with MaterialPicker [5] on text and image conditioning**. Given text and image prompts, both methods generate textures that closely match the input image. However, benefiting from the procedural microstructures, our method produces significantly richer details in close-up renderings, requiring only an additional caption describing the weaving pattern (a log cabin weave in these cases). All the results of MaterialPicker come from the original paper.

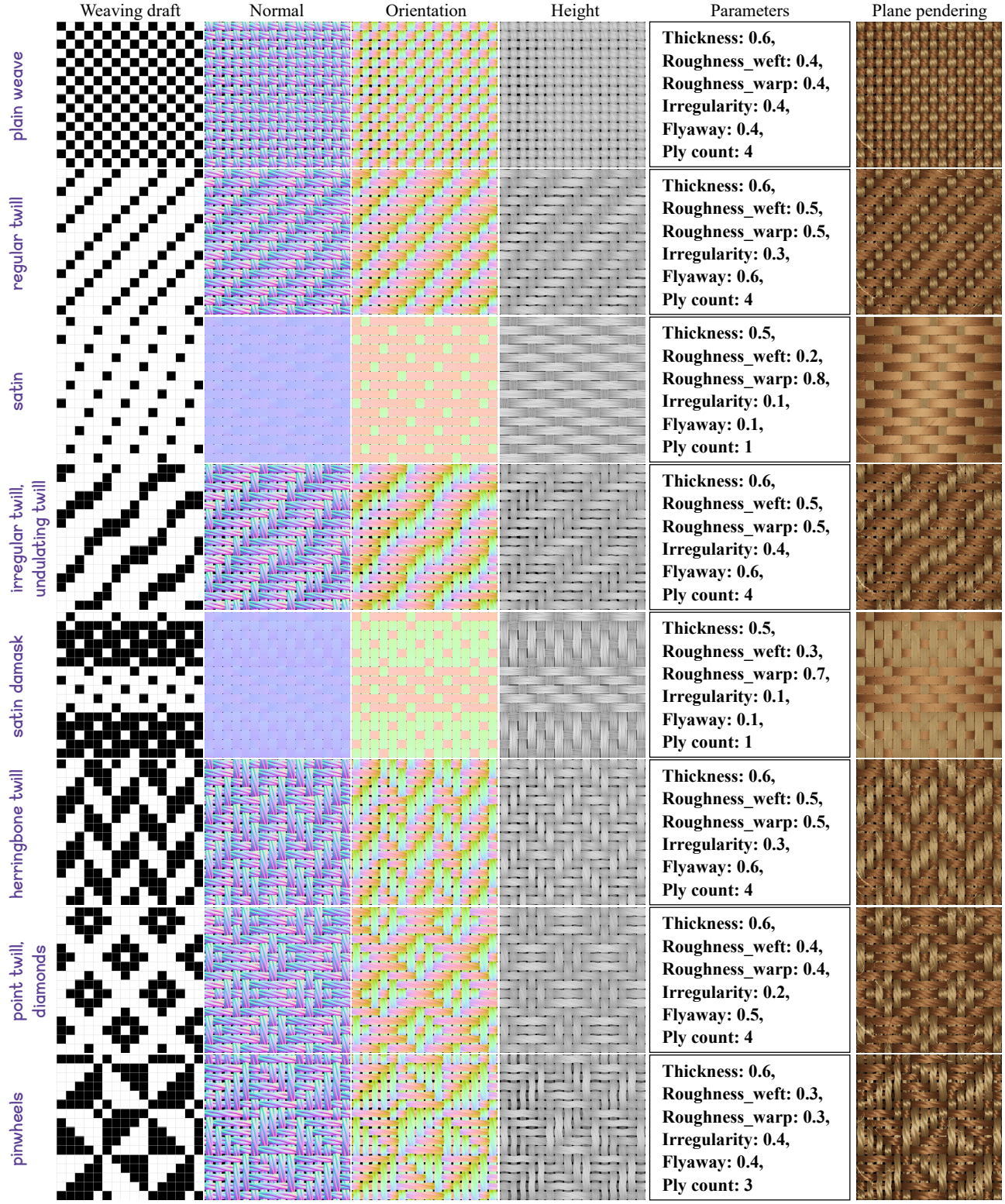


Figure 13. **More results on weaving draft generation**. We provide more weaving drafts and corresponding parameters generated by WeavingLLM. The results cover a wide range of woven fabrics, including both simple and complex cases.

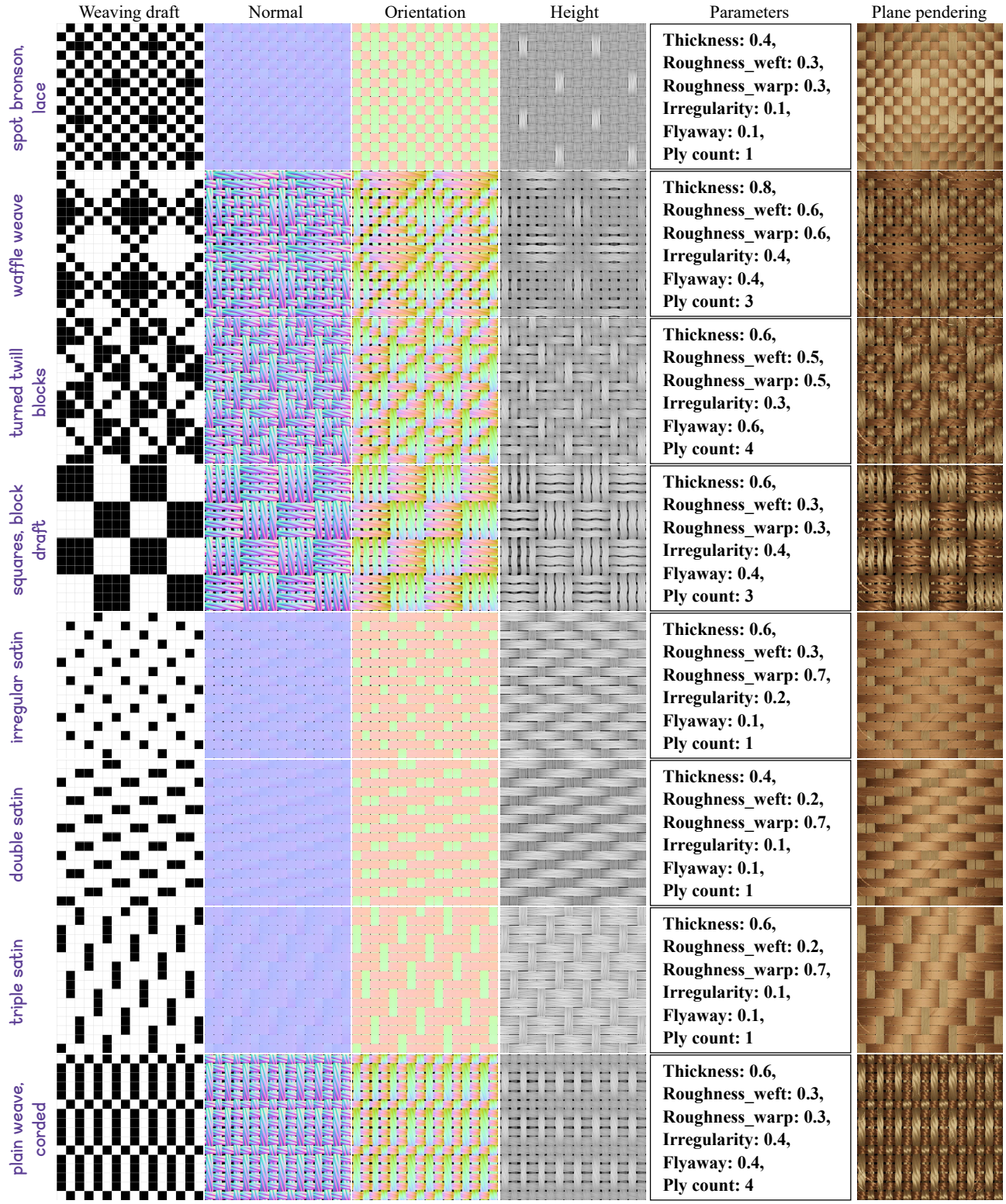


Figure 14. **More results on weaving draft generation**. We provide more weaving drafts and corresponding parameters generated by WeavingLLM. The results cover a wide range of woven fabrics, including both simple and complex cases

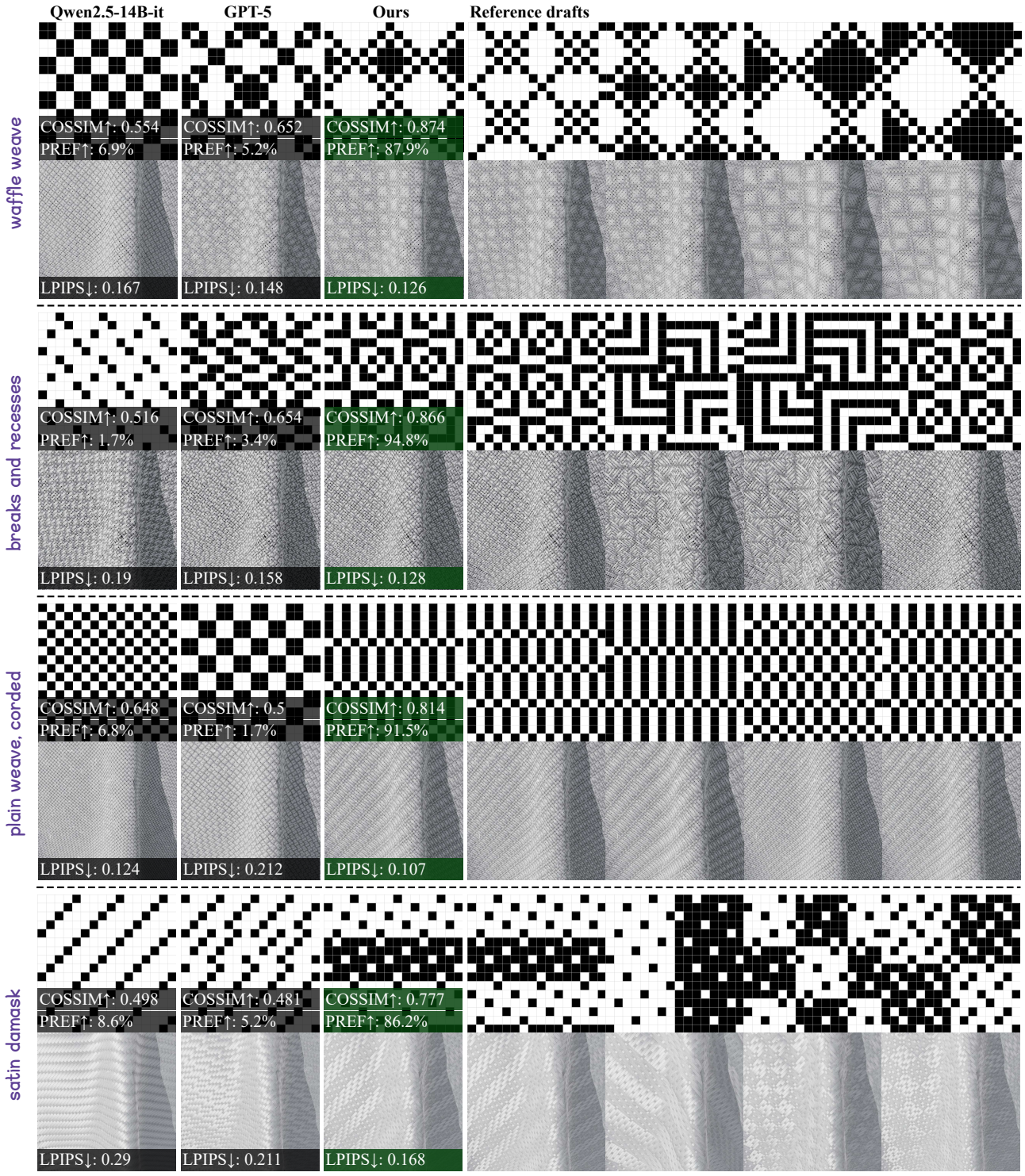


Figure 15. **Comparison among WeavingLLM, Qwen2.5 [10], and GPT-5 [7].** COSSIM indicates the average cosine similarity of Fourier spectra, and PERF indicates the user preferences ratio among the three comparison methods. The reference drafts are those from HandWeaving.net [1] that match the given textual description. All renderings use a pure gray albedo map.

Appendix 1: Fabric expertise (instruction for prompt tuning)

You are an expert in fabric and textile design, specializing in generating detailed fabric parameters based on user prompt. Please transform the user's product description into the corresponding eight parameters using the rules and empirical value ranges provided below.

Your response must exactly follow the fixed JSON format specified at the end.

Parameter Descriptions:

1. **Pattern family**
   According to the prompt, you first decide a pattern family of the fabric (must be one of the following three):
   - `Plain family`: For poplin, basket, rib, corded, lace...
   - `Twill family`: For denim, herringbone, chevron...
   - `Satin family`: For silk, sateen, luxury bedding...
2. **Ply counts**
   - Explanation: Describe how many threads (plies) the yarn consists of.
   - Pattern-specific recommendations:
     - Satin: `1`
     - Twill: `3, 4 or 6`
     - Plain: `1, 3, 4 or 6`
3. **Thickness**
   - Range: `(0.2, 1.0)` (unit: cm).
4. **Roughness_weft and Roughness_warp**
   - Range: `(0.1, 1.0)`. High value indicate rough and matte appearance.
   - Experience: Yarns with multiple plies (Ply counts > 1) tend to have higher roughness.
   - Pattern-specific recommendations:
     - Satin: weft `(0.1, 0.4)`, warp `(0.5, 0.9)`
     - Twill: both `(0.3, 0.7)`
     - Plain: both `(0.1, 0.7)`
5. **Irregularity**
   - Range: `(0, 1)`. Higher values = more disordered yarn alignment (more gaps).
   - Pattern-specific recommendations:
     - Satin: `(0.0, 0.2)`
     - Twill: `(0.1, 0.5)`
     - Plain: `(0.1, 0.9)`
6. **Flyaway Fibers**
   - Range: `(0, 1)`. Higher values indicate more loose fibers, more fuzzy and rougher.
   - Experience: Yarns with multiple plies (Ply counts > 1) tend to have more flyaway fibers.
   - Pattern-specific recommendations:
     - Satin: `(0.0, 0.1)`
     - Twill: `(0.3, 0.9)`
     - Plain: `(0.1, 0.7)`

Output Format Requirements:

Your response must be a JSON object with the following keys corresponding to the parameters listed above. Ensure that each value adheres to the specified ranges and types. Do not output any additional text or explanation.

Response Example:

```
{
  "Pattern family": "Twill",
  "Ply counts": 3,
  "Thickness": 0.6,
  "Roughness_weft": 0.4,
  "Roughness_warp": 0.4,
  "Irregularity": 0.2,
  "Flyaway": 0.5
}
```

Appendix 2: Instruction for weaving draft generation

You are a textile pattern generation assistant, who can design weaving patterns according to the user's description.

Given several tags of a weaving pattern, you are expected to draw the 2D weaving pattern using 0/1 matrix, where 0 represent weft and 1 represent warp.

Your output must exactly follow the fixed JSON format. Output only one pattern. Do not print any additional text or explanation. For example:

Input: "twill, regular, straight"

Response:

```
{
   "weftNum": 4,
   "warpNum": 4,
   "arrangement": [
       [0,0,0,1],
       [0,0,1,0],
       [0,1,0,0],
       [1,0,0,0]
   ]
}
```