Eurographics Symposium on Rendering 2022
A. Ghosh and L.-Y. Wei
(Guest Editors)

Point-Pattern Synthesis using Gabor and Random Filters: Supplemental Material

Xingchang Huang[1], Pooran Memari[2], Hans-Peter Seidel[1], Gurprit Singh[1]

[1]Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2]CNRS, LIX, Ecole Polytechnique, Paris, France

1. Analysis of Multi-scale Optimization

By-example point pattern synthesis methods usually requires users to tune with the window size or kernel size parameters as shown previous state-of-the-art methods [MWT11] [RÖM*15] [TLH19], to get satisfying synthesis results. Similarly, we use a multi-scale optimization strategy to preserve local and non-local structures for the synthesized patterns. Here we discuss the importance of multi-scale optimization and propose a way to tune with the two hyper-parameters which control the kernel size of our proposed Gabor features.

Single-scale vs. Multi-scale optimization. Fig. 1 demonstrates that our multi-scale optimization is important for preserving both global and local structures. The value of σ is analogous to receptive field, higher σ value capture more global structure while low σ focuses more on local structures. As shown in the figure, optimizing with only σ_1 results in a good global structure, but locally the points do not follow the regularity in the input. While optimizing only σ_2 leads to the missing global structure. Therefore, multi-scale optimization firstly focuses on synthesizing pattern with good global structure and then refines the local structures during the decrease of σ value.

Tuning Hyper-parameters. As discussed, a pattern can express different level of structures and choosing an appropriate window or kernel size is a necessary step for high-quality synthesis. In some cases, choosing the parameters wrongly can lead to unpleasing synthesis results. Therefore, we propose a pragmatic way to tune the two hyper-parameters, namely c_1 and c_2. As mentioned, most of the scenes use c_1 in 0.8 ± 0.2 and c_2 are 2.8 ± 0.2. For a new test scene, as default, we start from $c_1 = 0.6, c_2 = 2.6$. As shown in Fig. 2, when the number of points in the exemplar is small, the feature map of c_1 may not show the overall structure of the point pattern and lead to over-blurring features. This can result in less visually pleasing results. When we start increasing c_1 to be 0.6 and 0.8, the overall structure becomes more visible. Users can repeat this process until they are satisfied with the synthesized pattern. Fortunately, our method allows us to get satisfying synthesis results after few trials for most of the scenes. We summarize the parameters for all scenes in Table 1. As shown in the Table, the hyper-parameters are not so different across a large variety of scenes. This further demonstrates that our method is robust to hyper-parameter setting and user can find a good solution without too much manual effort.

Table 1: We summarize statistics and parameters used to synthesize each exemplar. M, S, R refer to the figure in main paper, supplemental material and the row number of each figure, respectively.

Scene	#Classes	#Output samples	Hyper-parameters	
			c_1	c_2
Fig. 5 (M), (a)	1	1124	1.0	3.0
Fig. 5 (M), (b)	1	256	1.0	2.6
Fig. 5 (M), (c)	1	992	0.8	2.6
Fig. 6 (M), (a)	2	208	0.8	2.6
Fig. 6 (M), (b)	2	512	1.6	2.6
Fig. 6 (M), (c)	4	216	1.0	2.8
Fig. 8 (M), R1	1	64	2.0	3.0
Fig. 8 (M), R2	2	512	1.6	2.6
Fig. 8 (S), R1	1	80	1.0	2.6
Fig. 8 (S), R2	1	192	0.8	2.6
Fig. 8 (S), R3	1	508	0.6	2.6
Fig. 8 (S), R4	1	472	0.8	2.6
Fig. 8 (S), R5	1	468	0.8	2.6
Fig. 9 (S), R1	1	424	1.0	2.6
Fig. 9 (S), R2	1	260	1.0	2.6
Fig. 9 (S), R3	1	476	0.8	2.6
Fig. 10 (S), R1	1	804	0.8	2.8
Fig. 10 (S), R2	1	1024	1.0	2.7

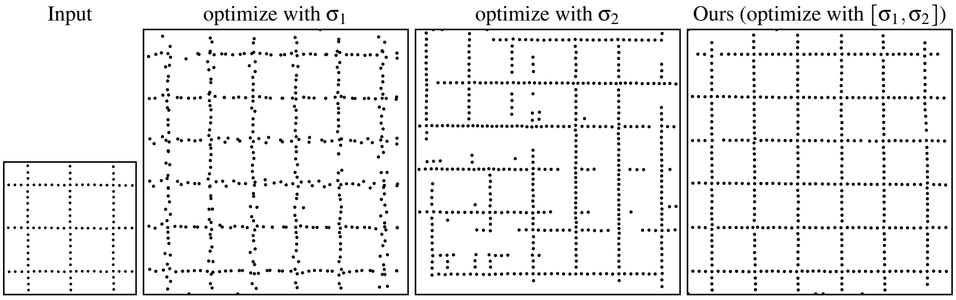
2. Diverse Results

We show that our method inherently supports synthesizing diverse results given the same exemplar, using randomly initialized point patterns with different random seeds. Fig. 3 shows three different outputs with different random initialization given the same input. Meanwhile, the overall structure looks similar to the exemplar. This also demonstrates that our method is robust to different initialization.

3. Additional Results

3.1. Ablation Study

Number of channels. In the convolutional filtering step, we define the number of output channels NC to control the number of random



(a) (b) (c) (d)

Figure 1: Importance of multi-scale optimization. If only optimized with a single scale, the synthesized results (b), (c) will only capture either global or local structure.

filters we use. Fig. 4 shows how the number of output channels in the convolutional layer affects the final results. We observe less accurate synthesis results with less filters defined by number of output channels in the convolutional filters. Increasing the number of output channels increases the synthesis quality. We find that $NC = 120$ is a reasonable choice as increasing the number of channels beyond 120 brings marginal difference while being more computationally expensive.

Layer for Correlation loss computation. We also study the layer noted as l chosen for computing $\mathcal{L}_{corr}$. We use the $4th$ ($l = 4$) layer for computing $\mathcal{L}_{corr}$. Other options are to use output from layer $l = 1, 2, 3$. However, $\mathcal{L}_{corr}$ becomes more computationally expensive with higher resolution feature maps. We observe out of memory issue while using the $1st$ or $2nd$ layer. Using the $3rd$ layer we get similar outputs as shown in Fig. 5, but the run-time is on average 6 times more than our choice.

3.2. Qualitative Comparisons

Point patterns. We show more results in Fig. 8 and Fig. 9 for qualitative comparisons between ours methods and previous state-of-the-art methods. Note that these patterns are included in the user study. Among them, our method achieves highest user scores compared with previous state-of-the-art methods on most of the scenes.

Element patterns. For discrete element-based pattern expansion, we experiment with different variants of the input exemplars and the method of [RGF*20]. Fig. 6 shows comparisons on 2-class patterns using our method, [RGF*20] and [TLH19]. As DiffCompositing [RGF*20] use only the Gram loss $\mathcal{L}_{gram}$ in their original paper, we test 2 more variants using their methods by including the Deep Correlation loss $\mathcal{L}_{corr}$. However, as shown in Fig. 7, we do not observe obvious improvement using their method, especially on the orange pattern with clear vertical structures in the middle column. On the other hand, our method performs better in terms of local and non-local structures. For patterns with more randomized structures, our method synthesizes patterns with less overlaps and closer to the exemplar's structure compared with [RGF*20].

Further, we show in Fig. 11 that our methods not only apply on discrete elements in $2D$, but also elements with higher-dimensional features. As shown in Fig. 11, our method takes input point patterns with features including depth, scale and $3D$ orientation for synthesis. This allows us to use the synthesized patterns for object placement and pattern design not limited to $2D$ space.

Table 2: We perform user study and compute an average score across 28 users. We show the average score for each pattern where 1 is the worst and 5 is the best. Our method gets better score for all but one pattern. M, S, R refer to the figures in main paper, supplemental material and row number of the corresponding figure, respectively.

Scene	User Scores (↑)			
	[MWT11]	[RÖM*15]	[TLH19]	Ours
Fig. 5 (M), (a)	1.4286	1.9643	3.6071	**4.2857**
Fig. 5 (M), (b)	3.7143	2.6071	3.5357	**4.6071**
Fig. 5 (M), (c)	1.4643	1.3571	3.6071	**4.4286**
Fig. 8 (S), R1	1.5714	3.0714	2.8571	**4.2857**
Fig. 8 (S), R2	1.2857	2.3214	4.2857	**4.3214**
Fig. 8 (S), R3	1.2857	1.9286	3.6071	**3.9286**
Fig. 8 (S), R4	2.5714	3.0357	4.0357	**4.6071**
Fig. 8 (S), R5	1.5000	3.9643	2.8214	**4.3929**
Fig. 9 (S), R1	1.5714	2.5357	3.6071	**4.3929**
Fig. 9 (S), R2	3.2143	3.0000	2.8929	**4.2500**
Fig. 9 (S), R3	1.5357	**4.5357**	3.4643	4.3214
Fig. 9 (S), R4	1.1071	3.3571	2.5000	**4.4643**
Fig. 10 (S), R1	1.2143	3.2143	2.8929	**4.1071**
Fig. 10 (S), R2	1.6071	2.7500	2.6429	**4.4643**

3.3. User Study

Table 3 shows the numbers of average scores from 28 participated users for the 14 point patterns used for our user study. Patterns we use are from Fig. 5 in the main paper, Fig. 8, Fig. 9 and Fig. 10 in the supplemental material.

References

[MWT11] Ma, Chongyang, Wei, Li-Yi, and Tong, Xin. "Discrete Element Textures". *ACM Trans. Graph.* 30.4 (July 2011). ISSN: 0730-0301. DOI: 10.1145/2010324.1964957 1, 2, 4, 6–8.

[RGF*20] Reddy, Pradyumna, Guerrero, Paul, Fisher, Matt, et al. "Discovering Pattern Structure Using Differentiable Compositing". *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: 10.1145/3414685.3417830 2, 5.

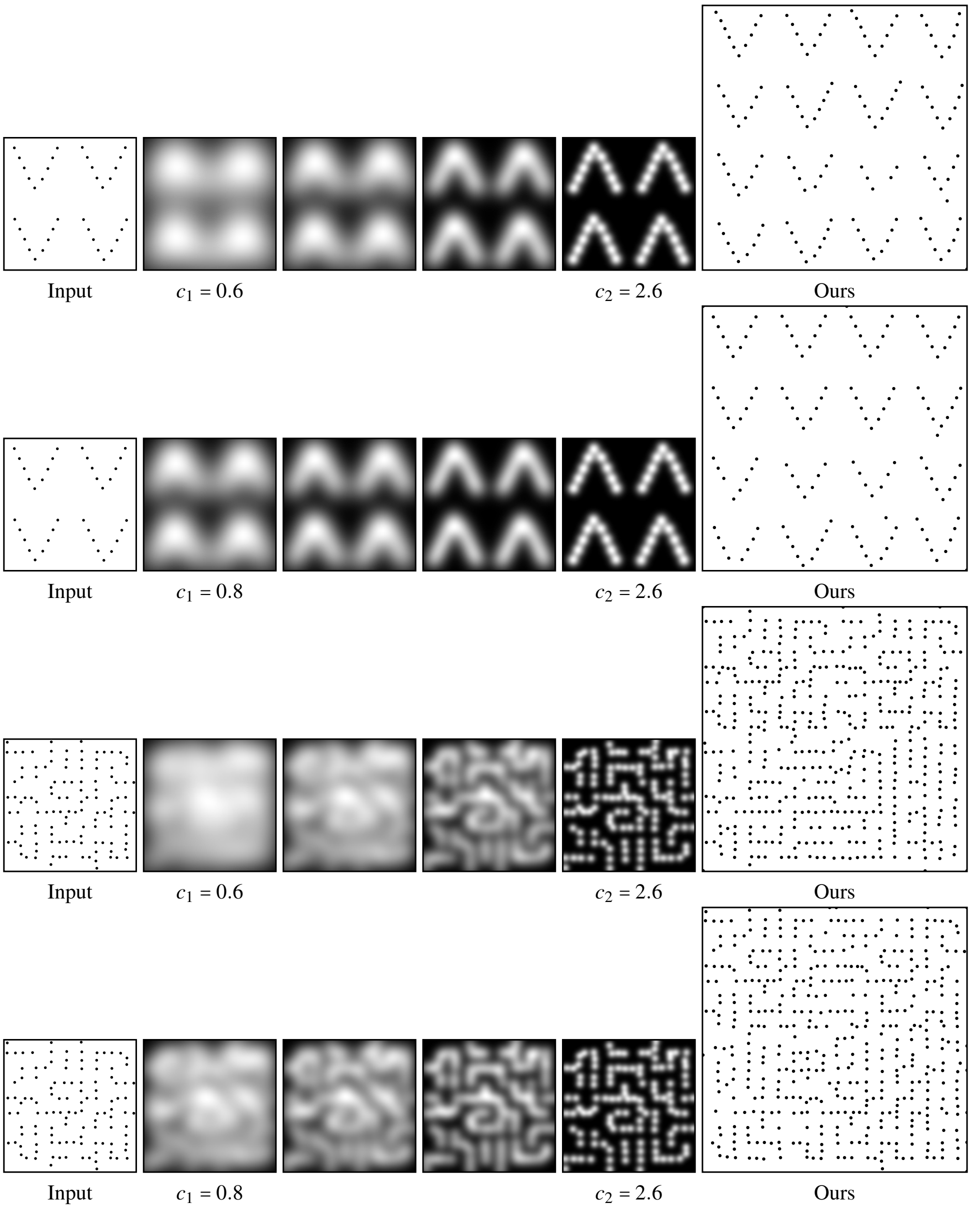


Figure 2: Hyper-parameters analysis. We demonstrate a pragmatic way to tune/increase the parameters c_1 and c_2 from the default values $c_1 = 0.6, c_2 = 2.6$ to get the final results.

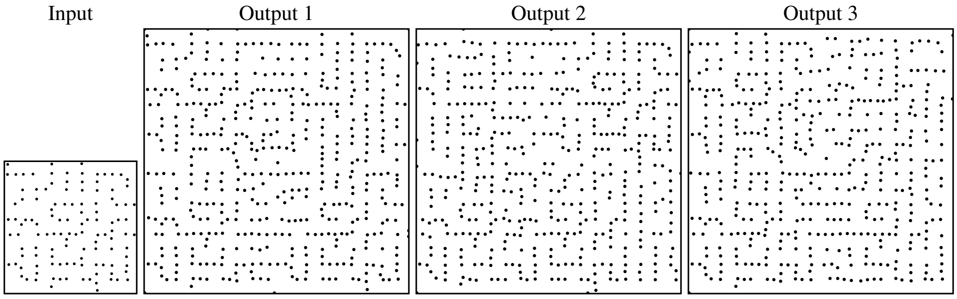


Figure 3: Diverse results. Given the same exemplar, our method synthesizes diverse outputs with different random initialization.

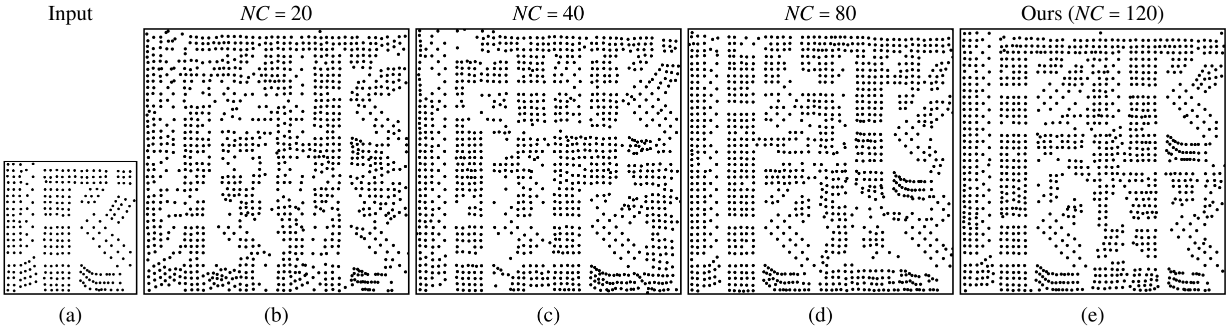


Figure 4: Ablation study on number of output channels (*NC*) in convolutional filters. Increasing *NC* leads to better results and $NC = 120$ is a reasonable choice considering the trade-off between run-time and synthesis quality.

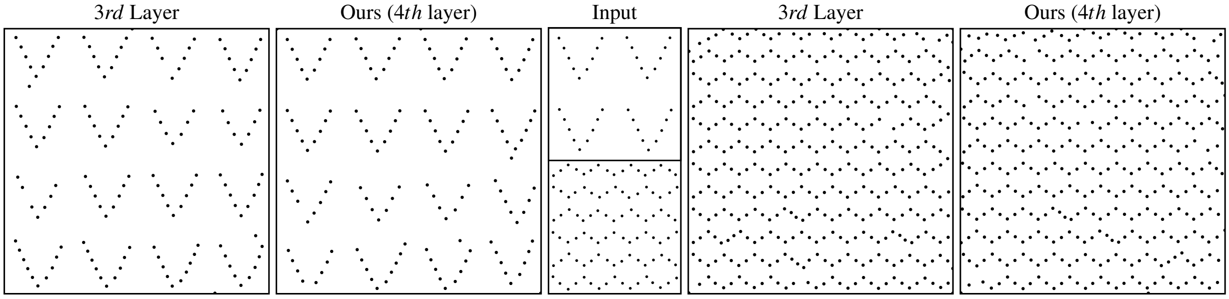


Figure 5: Ablation study on the layer selected for $\mathcal{L}_{corr}$ computation. Using the $4th$ layer output gives similar results compared with using the $3rd$ layer output, but is much less expensive due to lower resolution features.

Table 3: Quantitative comparisons for point pattern synthesis results of previous methods and ours. M, S, R refer to the figures in main paper, supplemental material and row number of the corresponding figure, respectively.

Scene	MSE of PCFs (↓)				Wasserstein Distance (↓)				Chamfer Distance (↓)			
	[MWT11]	[RÖM*15]	[TLH19]	Ours	[MWT11]	[RÖM*15]	[TLH19]	Ours	[MWT11]	[RÖM*15]	[TLH19]	Ours
Fig. 5 (M), (a)	0.2678	0.2836	0.2609	**0.2459**	1.4483	1.1148	1.0222	**0.6896**	0.0571	0.0440	0.0517	**0.0353**
Fig. 5 (M), (b)	0.3296	0.3286	0.3103	**0.3072**	0.4858	0.5166	0.4885	**0.3352**	0.0962	0.0869	0.0920	**0.0429**
Fig. 5 (M), (c)	0.6346	0.4913	0.5436	**0.3911**	2.1869	2.2596	2.2570	**0.5287**	0.0662	0.0685	0.0719	**0.0184**
Fig. 8 (S), R1	0.7562	0.7130	0.9043	**0.6395**	0.6292	0.6382	**0.6048**	0.7038	0.2131	0.1703	0.2116	**0.1601**
Fig. 8 (S), R2	0.4253	0.4188	0.3964	**0.3656**	0.7266	0.5322	0.4303	**0.2813**	0.1567	0.0701	0.0556	**0.0169**
Fig. 8 (S), R3	0.3731	0.3422	**0.2799**	0.2881	0.8372	0.7832	0.8406	**0.7158**	0.0868	0.0788	0.0905	**0.0774**
Fig. 8 (S), R4	0.2451	0.2642	0.2376	**0.2374**	0.6775	0.6040	0.6415	**0.5068**	0.0803	0.0685	0.0718	**0.0344**
Fig. 8 (S), R5	0.5038	0.3548	**0.3455**	0.3538	1.3144	**0.2885**	1.1597	0.3749	0.1000	0.0342	0.1233	**0.0271**
Fig. 9 (S), R1	0.5507	0.5537	0.4283	**0.4138**	1.3111	1.1803	1.3618	**0.8991**	0.1065	0.1123	0.1200	**0.0779**
Fig. 9 (S), R2	0.2915	0.2947	0.2858	**0.2786**	0.7385	0.7409	0.9657	**0.6828**	0.1221	0.0946	0.1205	**0.0990**
Fig. 9 (S), R3	0.2174	0.2570	**0.2403**	0.2427	1.1793	0.6341	0.7887	**0.6230**	0.1031	**0.0543**	0.0612	0.0631
Fig. 9 (S), R4	0.2661	0.2528	0.2623	**0.2483**	0.7412	0.5076	0.5882	**0.2367**	0.0746	0.0170	0.0548	**0.0149**
Fig. 10 (S), R1	0.4153	0.3438	**0.2872**	0.3429	0.9996	0.8678	0.8087	**0.6586**	0.0723	0.0394	0.0408	**0.0294**
Fig. 10 (S), R2	0.2401	0.2218	**0.2199**	0.2219	0.5531	0.3562	0.4701	**0.2345**	0.0473	**0.0043**	0.0312	0.0070

[RÖM*15] Roveri, Riccardo, Öztireli, A. Cengiz, Martin, Sebastian, et al. "Example Based Repetitive Structure Synthesis". *Proceedings of the Eurographics Symposium on Geometry Processing*. SGP '15. Graz, Austria: Eurographics Association, 2015, 39–52. doi: 10.1111/cgf.12695 1, 2, 4, 6–8.

[TLH19] Tu, Peihan, Lischinski, Dani, and Huang, Hui. "Point Pattern Synthesis via Irregular Convolution". *Computer Graphics Forum* 38.5 (2019), 109–122. doi: https://doi.org/10.1111/cgf.13793 1, 2, 4–8.

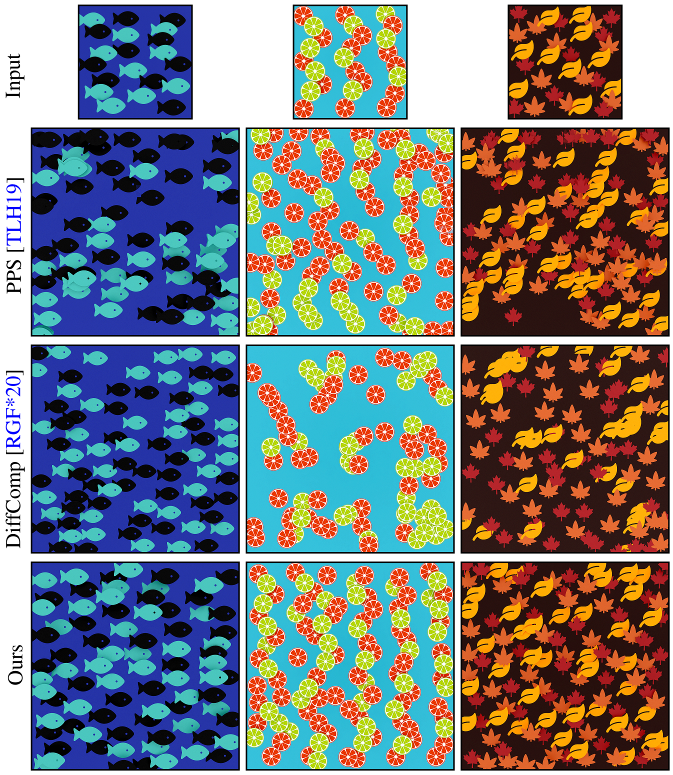


Figure 6: Discrete element-pattern expansion. We show 2-, 2- and 3-class examples from left to right and comparisons between [TLH19][RGF*20] and our method from top to bottom.

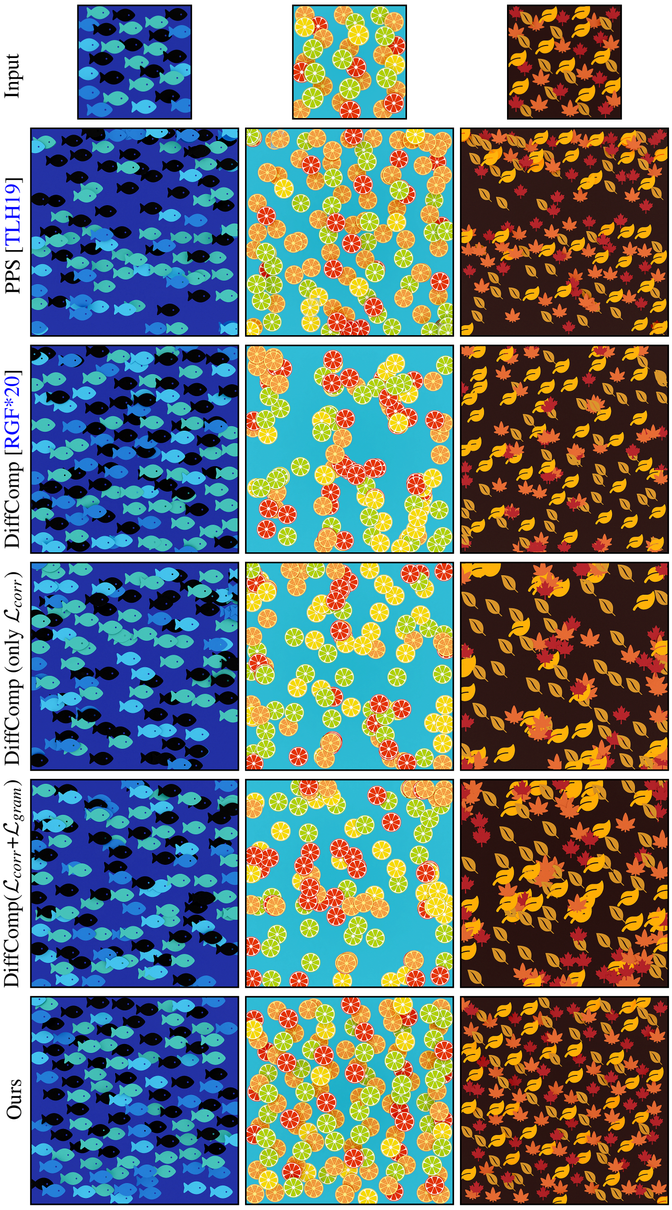

Input
PPS [TLH19]
DiffComp [RGF*20]
DiffComp (only $\mathcal{L}_{corr}$)
DiffComp($\mathcal{L}_{corr}$+$\mathcal{L}_{gram}$)
Ours

Figure 7: Discrete element-pattern expansion with 4-class examples. We test two more variants of [RGF*20] with only $\mathcal{L}_{corr}$ and a weighted combination of $\mathcal{L}_{corr}$ and $\mathcal{L}_{gram}$ for fair comparisons with [TLH19] and our method.

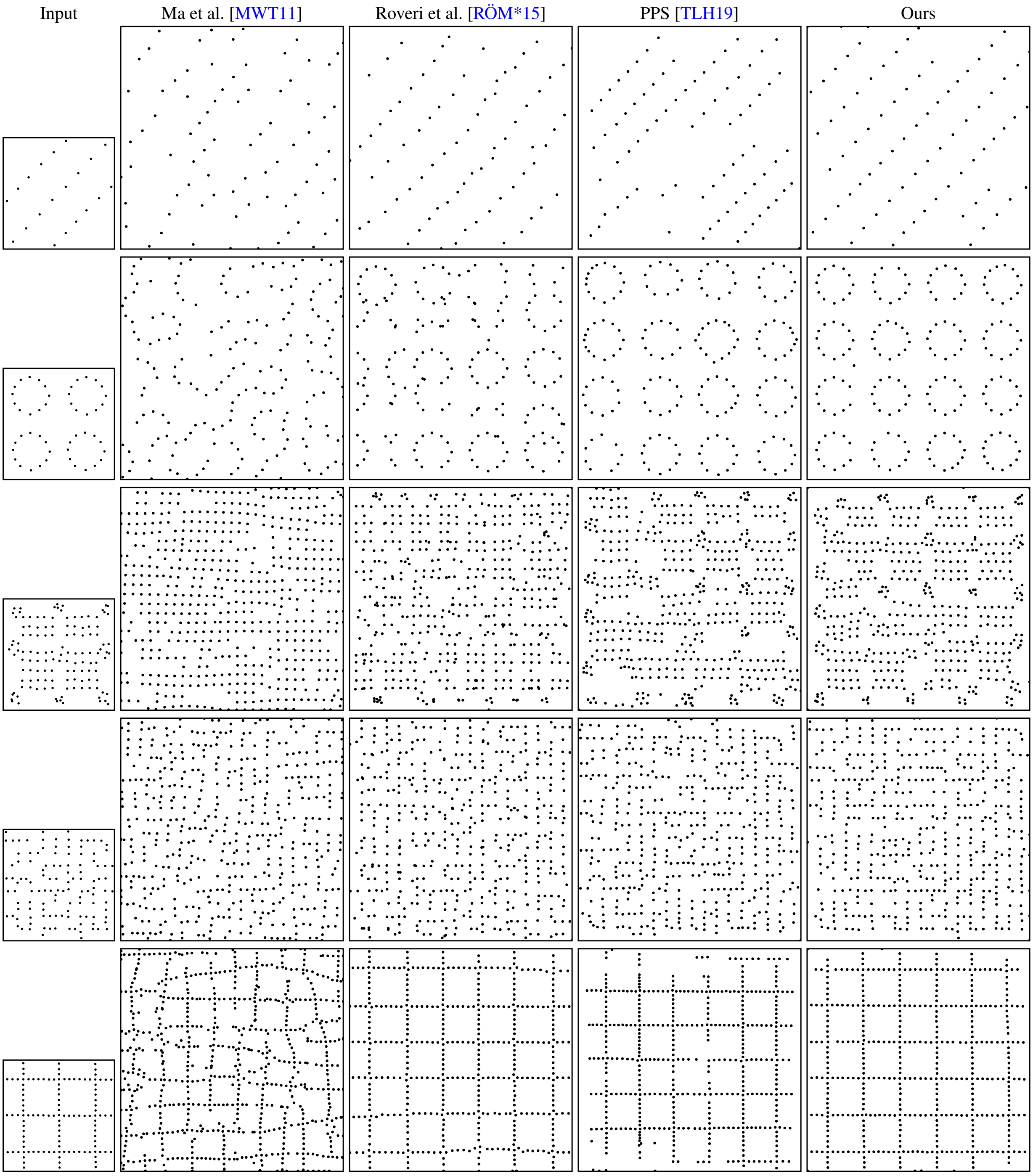


Figure 8: Additional comparisons between prior methods [MWT11], [RÖM*15], [TLH19] and ours, respectively.

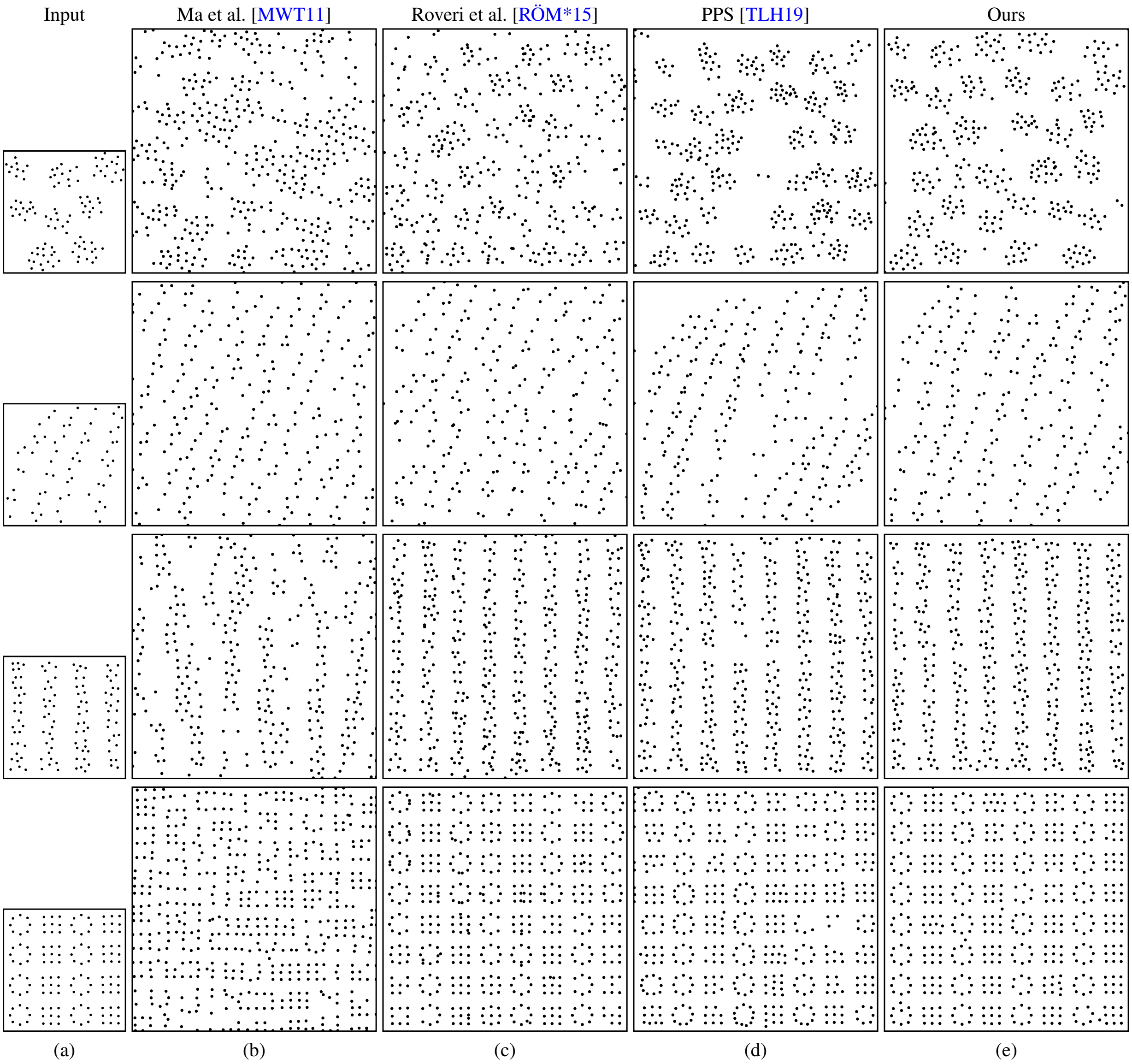


Figure 9: Additional comparisons between prior methods [MWT11], [RÖM*15], [TLH19] and ours, respectively.

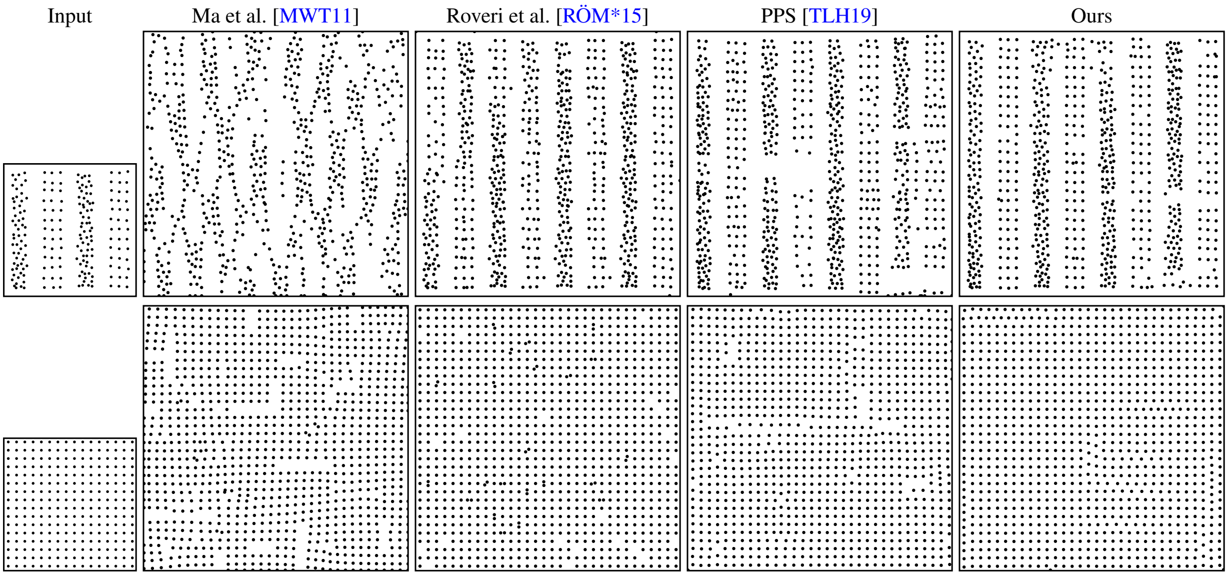
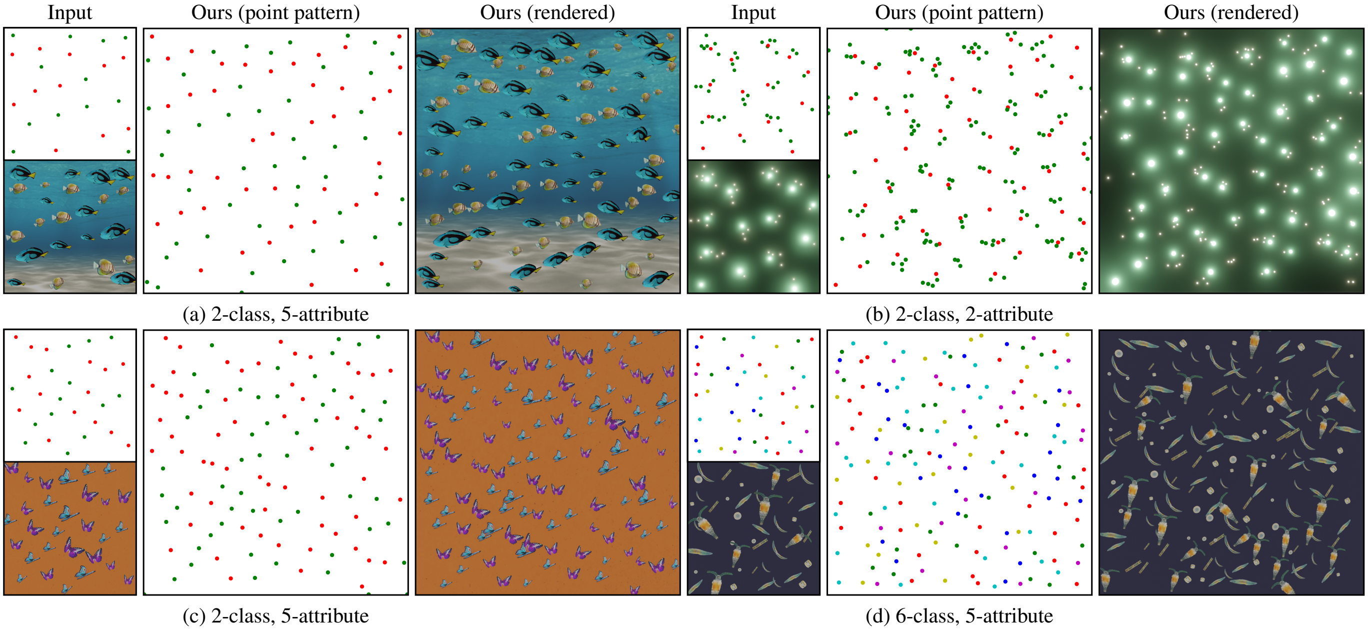


Figure 10: Additional comparisons between prior methods [MWT11], [RÖM*15], [TLH19] and ours, respectively.



(a) 2-class, 5-attribute

(b) 2-class, 2-attribute

(c) 2-class, 5-attribute

(d) 6-class, 5-attribute

Figure 11: Multi-attribute and multi-class point pattern synthesis results. We show 2-class examples from (a) to (c) and a 6-class example in (d). To visualize the patterns with multiple classes and attributes, we show $2D$ point patterns along with the rendering of 5 attributes including scale, depth and $3D$ orientation.