A Simple Push-Pull Algorithm for Blue-Noise Sampling

Abdalla G. M. Ahmed*, Jianwei Guo*, Dong-Ming Yan, Jean-Yves Franceschi, Xiaopeng Zhang, Oliver Deussen

In Section 1, we analyze the convergence behaviour of the Push-Pull algorithm. Then in Section 2, we provide a detailed datasheet to analyze the influence of different target parameters. We also compare the spectral and anti-aliasing properties of our method with that of alternative blue noise algorithms. Section 3 shows more surface remeshing results compared to previous work.

1 Convergence Analysis

This section exposes our observations concerning the convergence of our algorithm. Unfortunately, we could not find any theoretical results about its convergence behaviour (e.g., regarding its convergence zone or its speed of convergence); therefore, the following remarks only rely on experimental results, which provide however clear results.

Before beginning the convergence analysis, we will try to provide an intuition that could explain informally why our algorithm converges in most of the cases, based on the obervation of the evolution of point sets during its execution. Our algorithm is based on iteratively solving local constraints to finally obtain a point set that satisfies globally these constraints. The push-pull strategy allows, when constraints are not locally satisfiable, i.e., solving them for one point breaks these constraints for one of its neighbours, to randomly alter the structure of the point set and move these local constraints to another location where it will be easier to solve them. Thus, the algorithm globally changes the point set to another one satisfying the input conditions. In most of the cases, if the input constraints are not too strong, this random walk is quickly stopped because one of the numerous point sets satisfying these constraints has bee found. However, this does not hold if the input constraints are strong (e.g. for r_f close to 1), and we will try in the following to study the convergence behaviour in this case.

1.1 Definition of Convergence

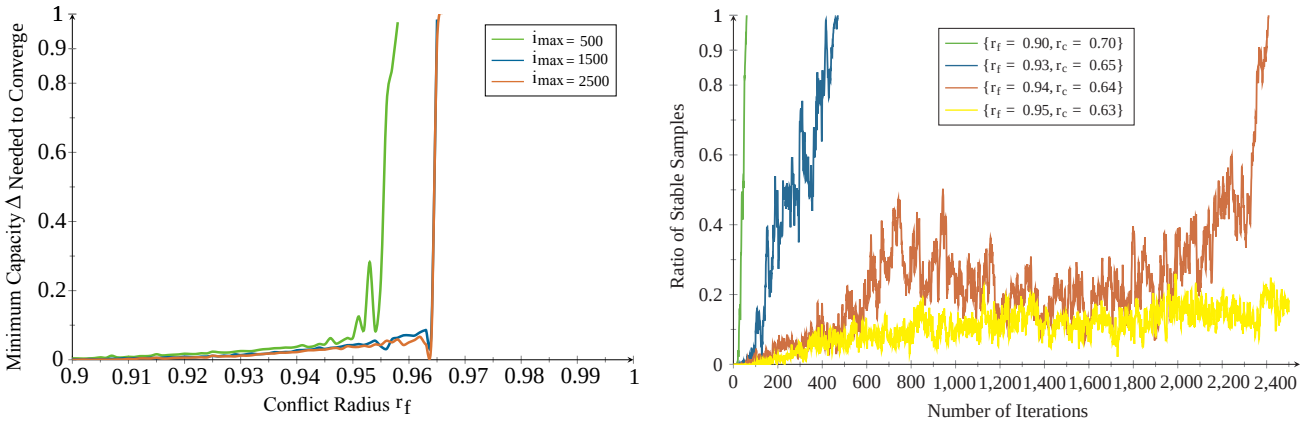
First, we have to find an experimental test that can attest that a configuration $(r_\mathrm{f}, r_\mathrm{c}, \Delta)$ makes the algorithm diverge. We define it for a given size of samples n: we say that a configuration is not achievable using our algorithm if it does not converge on this configuration until a given number of iterations i_max. i_max is chosen such that increasing it does not extend significantly the set of achievable configurations. For instance, for $n = 1024$, i_max can be set to 2500, as it will be shown later. The convergence zone, i.e., the set of achievable configurations, induced by this convergence definition is experimentally valid: a run (without limiting the number of iterations) of our algorithm on a non-achievable configuration will, in almost all cases, never converge.

1.2 Shape of the Convergence Zone

Experimentally, for a fixed Δ, it turns out that the convergence zone is a square: all configurations $(r_\mathrm{f}, r_\mathrm{c}, \Delta)$ are achievable if and only if $r_\mathrm{f} \leq r_\mathrm{f}^\mathrm{lim}$ and $r_\mathrm{c} \leq r_\mathrm{c}^\mathrm{lim}$. On the boundaries of this square, the convergence behaviour is erratic, no configuration that does not respect these constraints is achievable.

This observation implies that the convergence zone, if a parameter r that is either r_f or r_c is fixed, does not depend on the value of r, as long as this value is not close to r^lim for a Δ close to 0. The following study is done for a fixed $r_\mathrm{c} = 0.7$; the results are similar for a fixed $r_\mathrm{f} = 0.85$.

• *Joint first authors with equal contributions.



(a) The evolution of the minimum Δ that is required for the algorithm to converge.

(b) Convergence Speed.

Fig. 1. Convergence analysis of the proposed Push-Pull algorithm.

To study this profile, we computed the evolution of the minimum Δ that is required for the algorithm to converge, for $r_f \in [0.9, 1]$, as shown in Figure 1 (a), and we repeated this experiment for several i_{max}. It turns out that the minimum Δ required is very close to 0 for any r_f that is less than some r_f^{lim}, and almost instantly rise to 1 for $r_f = r_f^{lim}$ until the algorithm does not converge anymore for any given Δ.

These observations then provide a simple description of the convergence zone: all configurations with $r_f < 0.93$ and $r_c > 0.65$ are achievable; if only one of these thresholds is not satisfied, then the algorithm still converges as long as $r_f < 0.96$ and $r_c > 0.63$; if both are not satisfied, then the algorithm does not converge. Actually, the convergence behaviour does depend on Δ, but the previous observations tend to show that the importance of this parameter is negligible.

Please notice that r_f^{lim} is, as expected, an increasing funtion of i_{max}; we can notice that, for i_{max} high enough, $r_f^{lim} \approx 0.965$, which experimentally validates our convergence definition that relies on an invariance of the convergence zone once i_{max} goes past some high enough value.

1.3 Convergence Speed at the Boundaries of the Convergence Zone

What remains to do is to study how our algorithm behaves when r_f is close to 1, or r_c is close to 0.6. We performed several experiments for a fixed Δ that are shown in Figure 1 (b). Our observations indicate that all configurations with $r_f < 0.9$ and $r_c > 0.67$ are achievable in around a hundred of iterations. If one of those two parameters goes past its threshold, the number of iterations required for convergence increases very quickly (from a hundred to 2500 for 1024 points), until the algorithm does not converge anymore. In this case, the ratio of stable samples during the execution grows slowly until it remains stable.

1.4 Relevance of the Combination Order of the Optimizations

The previous tests were done using a fixed combination of the three possible optimization algorithm (coverage, conflict, and capacity optimization, in this order). It is worth noticing that the results obtained with other combinations are extremely similar; it is therefore safe to state that the convergence zone of our algorithm as well as its time efficiency is not very dependent on the way the optimizations are combined.

2 Parameter Selection and Comparison

For comparison, we first give the mostly used spatial measures of state-of-the-art blue noise algorithms in Table 1. Then we test our algorithm by varying the parameters r_c and r_f (Δ is always set as 0.0386) in Table 2. Figures 2 and 3 show the power spectra and zone plate plots of other methods and our algorithm with fixed $r_c = 0.67$, respectively.

Other Methods	$d_{\min}$	d_{avg}	r_{c}	β	ν_{eff}	Ω	BOO
White	0.023	0.465	1.635	71.420	0.0	0.074	0.363
MPS [1]	0.780	0.821	0.781	1.0	0.655	2.168	0.388
BNOT [2]	0.738	0.872	0.744	1.008	0.855	1.994	0.427
CVT [3]	0.805	0.941	0.657	0.817	0.950	4.733	0.842
CCVT [4]	0.711	0.856	0.782	1.10	0.830	1.774	0.403
CapCVT [5]	0.755	0.895	0.772	1.022	0.905	2.957	0.661
KDM [6]	0.645	0.868	0.746	1.157	0.870	2.223	0.451
FPO [7]	0.925	0.933	0.869	0.939	0.900	4.628	0.428

TABLE 1
Statistics of the popular spatial measures of previous blue noise sampling patterns.

Our target parameters		Results						
r_{c}	r_{f}	d_{min}	d_{avg}	r_{c}	β	ν_{eff}	Ω	BOO
0.63	0.65	0.655	0.826	0.630	0.962	0.835	1.653	0.466
	0.70	0.700	0.828	0.630	0.900	0.835	1.671	0.470
	0.75	0.755	0.835	0.630	0.840	0.845	1.744	0.470
	0.80	0.800	0.847	0.630	0.787	0.860	1.894	0.487
	0.85	0.850	0.870	0.630	0.741	0.880	2.290	0.499
	0.90	0.900	0.905	0.630	0.700	0.905	3.189	0.541
	0.93	0.930	0.932	0.630	0.677	0.920	4.077	0.593
0.65	0.65	0.650	0.807	0.650	1.000	0.785	1.283	0.426
	0.70	0.700	0.814	0.650	0.928	0.795	1.349	0.433
	0.75	0.750	0.823	0.650	0.867	0.810	1.448	0.437
	0.80	0.800	0.839	0.650	0.812	0.830	1.652	0.448
	0.85	0.850	0.867	0.650	0.765	0.860	2.157	0.474
	0.90	0.900	0.905	0.650	0.722	0.895	3.180	0.499
	0.93	0.930	0.932	0.650	0.699	0.915	4.156	0.554
0.67	0.65	0.651	0.796	0.670	1.030	0.750	1.085	0.410
	0.70	0.700	0.809	0.670	0.957	0.770	1.225	0.427
	0.75	0.750	0.814	0.670	0.893	0.780	1.261	0.422
	0.80	0.800	0.835	0.670	0.837	0.805	1.561	0.427
	0.85	0.850	0.866	0.670	0.788	0.845	2.128	0.450
	0.90	0.900	0.905	0.670	0.744	0.885	3.178	0.482
	0.93	0.930	0.932	0.670	0.720	0.910	4.207	0.521
0.70	0.65	0.650	0.780	0.700	1.076	0.705	0.848	0.386
	0.70	0.700	0.797	0.700	1.000	0.725	0.996	0.402
	0.75	0.750	0.812	0.700	0.933	0.750	1.200	0.412
	0.80	0.800	0.833	0.700	0.875	0.780	1.536	0.416
	0.85	0.850	0.865	0.700	0.823	0.830	2.135	0.436
	0.90	0.900	0.904	0.700	0.778	0.875	3.222	0.466
	0.93	0.930	0.932	0.700	0.753	0.905	4.231	0.515

TABLE 2
Statistics of the popular spatial measures of our methods with different parameter selections.

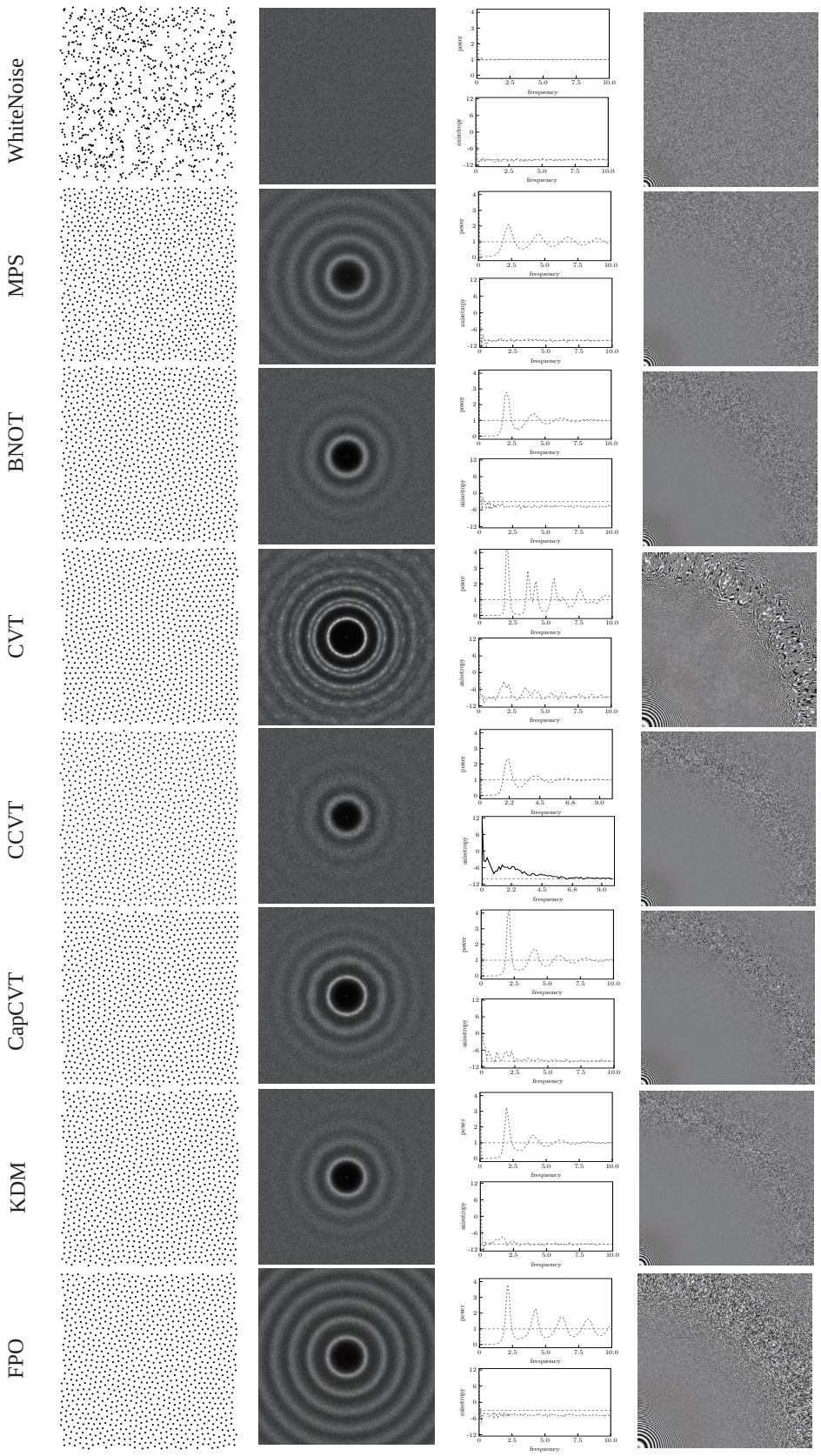


Fig. 2. Spectral and anti-aliasing analysis of previous blue noise methods. From left to right: distribution of 1024 points, power spectrum, radial means (top) and anisotropy (bottom), zone plate test function sampled by 512^2 points.

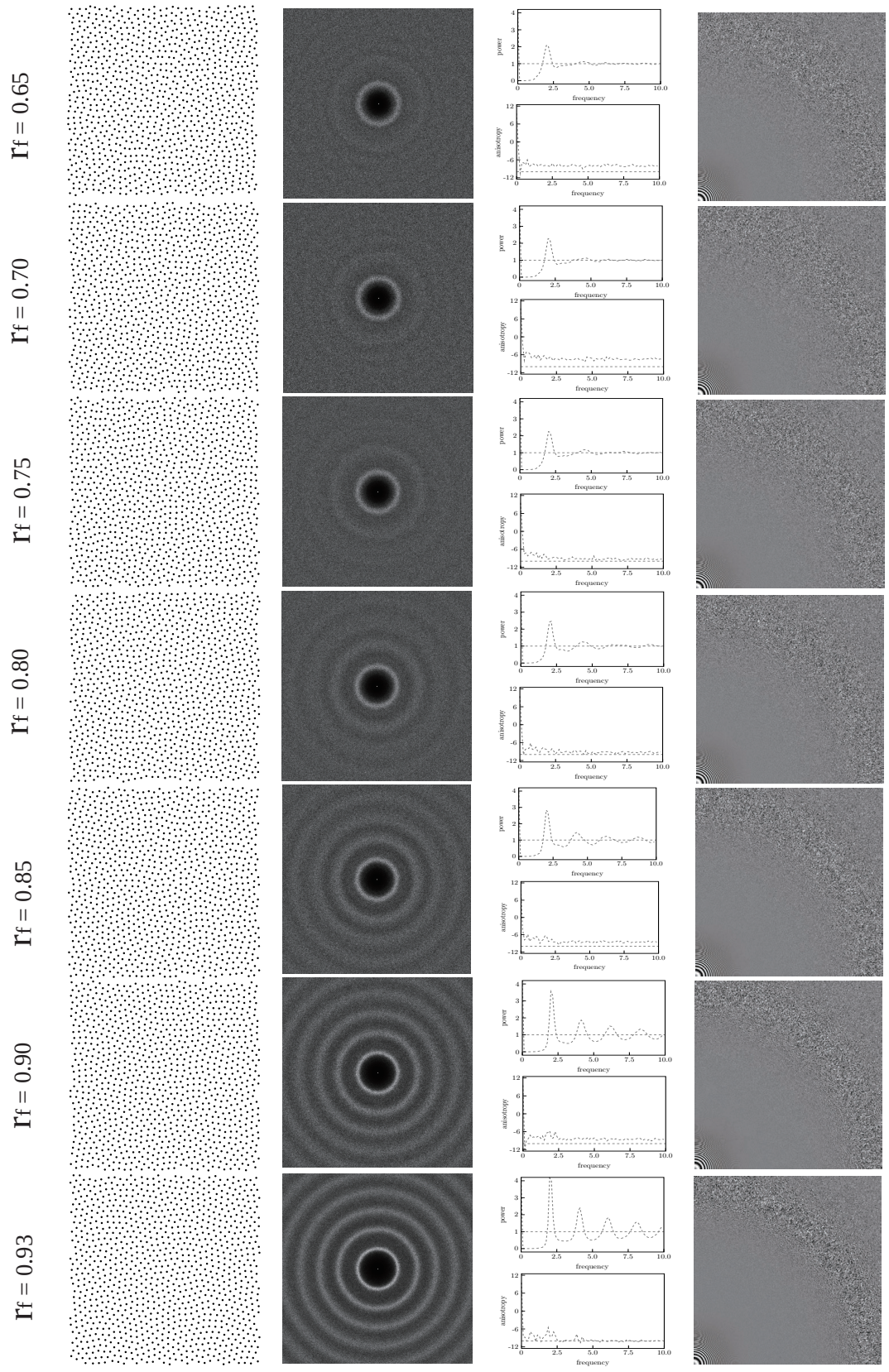


Fig. 3. Spectral and anti-aliasing analysis of our algorithm with fixed $r_{\mathrm{c}} = 0.67$. From left to right: distribution of 1024 points, power spectrum, radial means (top) and anisotropy (bottom), zone plate test function sampled by 512^2 points.

3 REMESHING EXPERIMENTS

In this section, we present surface remeshing comparisons between our push-pull algorithm and a wide range of other remeshing techniques, including: *Approximate Centroidal Voronoi Diagram* (ACVD) [8], *Maximal Poisson-disk Sampling* (MPS) [9] [10], *Farthest Point Optimization* (FPO) [11], *Capacity Constrained CVT* (CapCVT) [5], *Non-obtuse CVT* (CVT_{nob}) [12], *Disk Density Tuning of MPS* (DiskTuning) [13]. Table 3 lists the numerical statistics of remeshing quality compared with previous methods. Figures 4 to 10 visualize the remeshing results of all the models.

REFERENCES

[1] M. S. Ebeida, S. A. Mitchell, A. Patney, A. A. Davidson, and J. D. Owens, "A simple algorithm for maximal Poisson-disk sampling in high dimensions," *Computer Graphics Forum (Proc. EUROGRAPHICS)*, vol. 31, no. 2, pp. 785–794, 2012.

[2] F. de Goes, K. Breeden, V. Ostromoukhov, and M. Desbrun, "Blue Noise through Optimal Transport," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 31, pp. 171:1–171:12, 2012.

[3] S. A. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[4] M. Balzer, T. Schlömer, and O. Deussen, "Capacity-constrained point distributions: A variant of Lloyd's method," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 6, pp. 86:1–86:8, 2009.

[5] Z. Chen, Z. Yuan, Y.-K. Choi, L. Liu, and W. Wang, "Variational Blue Noise Sampling," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 18, no. 10, pp. 1784–1796, 2012.

[6] R. Fattal, "Blue-noise point sampling using kernel density model," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 3, pp. 48:1–48:10, 2011.

[7] T. Schlömer, D. Heck, and O. Deussen, "Farthest-point optimized point sets with maximized minimum distance," in *High Performance Graphics Proceedings*, 2011, pp. 135–142.

[8] B. Lévy and N. Bonneel, "Variational anisotropic surface meshing with Voronoi parallel linear enumeration," in *Proceedings of the 21st International Meshing Roundtable*, 2012, pp. 349–366.

[9] D.-M. Yan and P. Wonka, "Gap processing for adaptive maximal Poisson-disk sampling," *ACM Trans. on Graphics*, vol. 32, no. 5, pp. 148:1–148:15, 2013.

[10] J. Guo, D.-M. Yan, X. Jia, and X. Zhang, "Efficient maximal Poisson-disk sampling and remeshing on surfaces," *Computers & Graphics (Proc. SMI)*, vol. 46, no. 6-8, pp. 72–79, 2015.

[11] D.-M. Yan, J. Guo, X. Jia, X. Zhang, and P. Wonka, "Blue-noise remeshing with farthest point optimization," *Computer Graphics Forum (Proc. SGP)*, vol. 33, no. 5, pp. 167–176, 2014.

[12] D. Yan and P. Wonka, "Non-obtuse Remeshing with Centroidal Voronoi Tessellation," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 22, no. 9, pp. 2136–2144, 2016.

[13] M. Ebeida, A. A. Rushdi, M. A. Awad, A. H. Mahmoud, D.-M. Yan, S. A. English, J. D. Owens, C. L. Bajaj, and S. A. Mitchell, "Disk density tuning of a maximal random packing," *Computer Graphics Forum (Proc. SGP)*, 2016.

[14] P. Frey and H. Borouchaki, "Surface mesh evaluation," in *6th Intl. Meshing Roundtable*, 1997, pp. 363–374.

[15] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998.

Model	Method	$\|\mathbf{X}\|$	$\|t\|$	$\|t_{obt}\|$	Q_{min}	Q_{avg}	θ_{min}	$\bar{\theta}_{min}$	θ_{max}	$\theta_{<30^o}\%$	$\theta_{>90^o}\%$	$V_{567}\%$	$d_{RMS}(\times 10^{-3})$	$d_H(\times 10^{-2})$
Venus	ACVD	3.0K	6.0K	466	0.25	0.84	14.4	47.6	146.3	0.36	7.77	98.4	0.86	0.69
	MPS	3.0K	6.0K	**0**	0.67	0.85	32.7	48.6	90.0	**0**	**0**	**100**	0.73	0.67
	FPO	3.0K	6.0K	377	0.57	0.85	34.2	50.8	107.1	**0**	6.29	99.7	0.71	0.67
	CapCVT	3.0K	6.0K	1.0K	0.39	0.78	20.5	43.3	128.9	4.41	17.7	98.8	0.75	0.57
	CVT	3.0K	6.0K	15	0.65	0.93	39.5	54.5	97.3	**0**	0.25	**100**	0.84	0.56
	CVT_{nob}	3.0K	6.0K	**0**	0.73	**0.96**	37.4	**55.9**	**86.1**	**0**	**0**	**100**	0.68	**0.53**
	Our	3.0K	6.0K	**0**	**0.74**	0.87	**44.6**	50.9	87.9	**0**	**0**	**100**	**0.60**	0.56
Genus	ACVD	6.5K	13K	913	0.47	0.85	29.0	48.1	118.6	0.06	7.02	98.1	0.71	0.42
	MPS	6.5K	13K	**0**	0.66	0.85	31.3	48.5	90.0	**0**	**0**	**100**	0.67	0.60
	FPO	6.5K	12K	737	0.55	0.86	33.1	50.9	109.3	**0**	6.14	99.6	0.69	0.56
	CapCVT	6.5K	12.5K	2.3K	0.38	0.78	16.8	42.6	130.0	5.91	18.3	98.6	0.64	0.61
	CVT	6.5K	13K	27	0.66	0.94	38.7	54.6	96.8	**0**	0.21	**100**	0.76	0.49
	CVT_{nob}	6.5K	13K	**0**	0.72	**0.95**	36.0	**55.6**	**88.9**	**0**	**0**	**100**	**0.31**	**0.23**
	Our	6.5K	13K	**0**	**0.73**	0.87	**44.2**	50.8	**88.9**	**0**	**0**	**100**	0.43	0.48
Fertility	MPS	8.5K	17K	2.6K	0.50	0.81	30.26	45.26	116.0	**0**	15.1	96.4	0.48	0.41
	FPO	8.5K	17K	1.0K	0.52	0.86	32.8	50.8	113.6	**0**	6.16	99.5	0.98	0.33
	CapCVT	8.5K	17K	508	0.52	0.88	30.02	51.3	113.5	**0**	2.98	99.8	0.42	**0.28**
	CVT	8.5K	17K	**0**	0.66	**0.94**	40.2	**54.9**	96.4	**0**	0.05	**100**	0.41	0.29
	CVT_{nob}	8.5K	17K	**0**	**0.77**	**0.94**	40.4	54.8	**83.4**	**0**	**0**	**100**	0.98	0.34
	DiskTuning	7.7K	15.5K	**0**	0.64	0.82	30.1	45.7	90.0	**0**	**0**	97.8	0.55	0.37
	Our	8.5K	17K	**0**	0.74	0.87	**46.3**	51.1	87.5	**0**	**0**	**100**	**0.29**	0.31
Moai	MPS	12K	24K	3.6K	0.47	0.81	30.4	45.3	118.8	**0**	14.8	96.1	0.91	0.54
	FPO	11K	22K	1.4K	0.54	0.86	32.6	50.9	110.3	**0**	6.08	99.5	0.56	0.51
	CapCVT	11K	2.2K	2.3K	0.48	0.82	21.1	46.4	118.5	0.65	10.4	99.4	0.55	0.45
	CVT	11K	22K	37	0.65	0.93	37.8	54.3	97.3	**0**	0.17	99.9	0.52	**0.39**
	CVT_{nob}	11K	22K	**0**	0.68	**0.95**	32.4	**55.8**	**85.2**	**0**	**0**	**100**	0.88	0.51
	DiskTuning	11K	22K	3	0.64	0.82	30.1	45.6	90.0	**0**	0.01	97.8	0.95	0.60
	Our	11K	22K	**0**	**0.74**	0.87	**45.6**	51.1	87.7	**0**	**0**	99.9	**0.43**	0.46
Homer	ACVD	7.5K	15K	2.7K	0.04	0.80	2.11	44.4	174.1	6.18	17.7	94.4	0.82	0.31
	MPS	7.5K	15K	1.6K	0.56	0.82	30.1	46.4	105.8	**0**	10.8	99.8	0.61	0.28
	FPO	7.7K	15.6K	1.2K	0.49	0.85	31.0	49.6	117.5	**0**	7.99	99.2	0.45	0.30
	CapCVT	7,5K	15K	1.6K	0.37	0.83	21.3	46.2	131.1	0.98	10.66	95.8	0.42	0.20
	CVT	7.5K	15K	56	0.65	0.93	**38.1**	54.0	98.2	**0**	0.37	**100**	0.57	0.20
	CVT_{nob}	7.5K	15K	**0**	**0.69**	**0.94**	33.7	**54.5**	**85.4**	**0**	**0**	**100**	**0.36**	**0.17**
	DiskTuning	3.8K	7.8K	1	0.51	0.82	21.3	45.0	90	1.64	0.01	**97.3**	1.21	0.57
	Our	7.5K	15K	**0**	0.68	0.90	34.4	51.8	89.5	**0**	**0**	**100**	0.40	0.23
Bunny	ACVD	8.0K	16K	3.3K	0.10	0.77	4.76	42.4	165.9	9.96	20.5	92.2	0.97	0.55
	MPS	7.8K	15.4K	755	0.62	0.83	32.0	47.4	97.9	**0**	4.89	99.8	0.77	0.46
	FPO	8.0K	16K	1.6K	0.50	0.84	30.0	48.6	116.3	**0**	9.92	98.0	0.81	0.42
	CapCVT	8.0K	16K	1.4K	0.39	0.84	15.6	47.3	125.1	1.04	8.46	98.7	0.71	**0.24**
	CVT	8.0K	16K	10	0.64	0.93	34.8	54.2	98.8	**0**	0.06	99.9	0.87	0.35
	CVT_{nob}	8.0K	16K	**0**	**0.72**	**0.94**	36.3	**54.4**	**89.4**	**0**	**0**	**100**	0.94	0.35
	DiskTuning	5.8K	12K	**0**	0.50	0.82	19.89	45.2	90.0	0.88	**0**	97.8	1.55	0.52
	Our	8.0K	16K	**0**	0.71	0.87	**37.1**	51.1	89.5	**0**	**0**	99.8	**0.44**	0.52
Kitten	ACVD	9.8K	16K	3.3K	0.10	0.77	4.76	42.4	165.9	9.96	20.5	92.2	0.61	0.59
	MPS	7.9K	15.8K	2.4K	0.42	0.81	25.3	45.3	125.4	0.29	15.3	96.0	0.78	0.96
	FPO	9.5K	19K	1.3K	0.53	0.85	32.0	50.0	111.6	**0**	6.89	99.4	0.53	0.29
	CapCVT	9.8K	19.6K	1.4K	0.39	0.84	15.6	47.3	125.1	1.04	8.46	98.7	0.29	0.21
	CVT	9.8K	19.6K	6	0.67	0.94	**39.2**	54.8	95.6	**0**	0.03	99.9	0.39	**0.20**
	CVT_{nob}	9.8K	19.6K	**0**	**0.71**	**0.95**	33.6	**55.3**	**87.1**	**0**	**0**	**100**	0.27	**0.20**
	Our	9.8K	19.6K	**0**	**0.71**	0.90	37.3	52.1	88.9	**0**	**0**	**100**	**0.26**	0.27

TABLE 3

Comparison of remeshing quality with previous techniques. The best result of each measurement is marked in **bold** font. $|\mathbf{X}|$ is the number of vertices; $|t|$ is the number of triangles; $|t_{obt}|$ is the number of obtuse triangles; Q_{min} is the minimal triangle quality, where the quality of a triangle is $Q(\mathbf{t}) = \frac{6}{\sqrt{3}} \frac{S_\mathbf{t}}{p_\mathbf{t} h_\mathbf{t}}$, where $S_\mathbf{t}$ is the area of $\mathbf{t}$, $p_\mathbf{t}$ is the half-perimeter of $\mathbf{t}$ and $h_\mathbf{t}$ the the longest edge length of $\mathbf{t}$ [14]; Q_{avg} is the average of the triangle qualities; θ_{min} is the minimal angle; $\bar{\theta}_{min}$ is the average of minimal angle of each triangle; θ_{max} is the maximal angle, $\theta_{<30^o}\%$ is the percentage of triangles with angles smaller than 30^o; $\theta_{>90^o}\%$ is the percentage of obtuse triangles; $V_{567}\%$ is the percentage of the valence 5, 6, and 7 vertices; d_{RMS} is the root mean square distance, and d_H is the Hausdorff distance between the remesh and the input surface, which is measured by Metro [15].

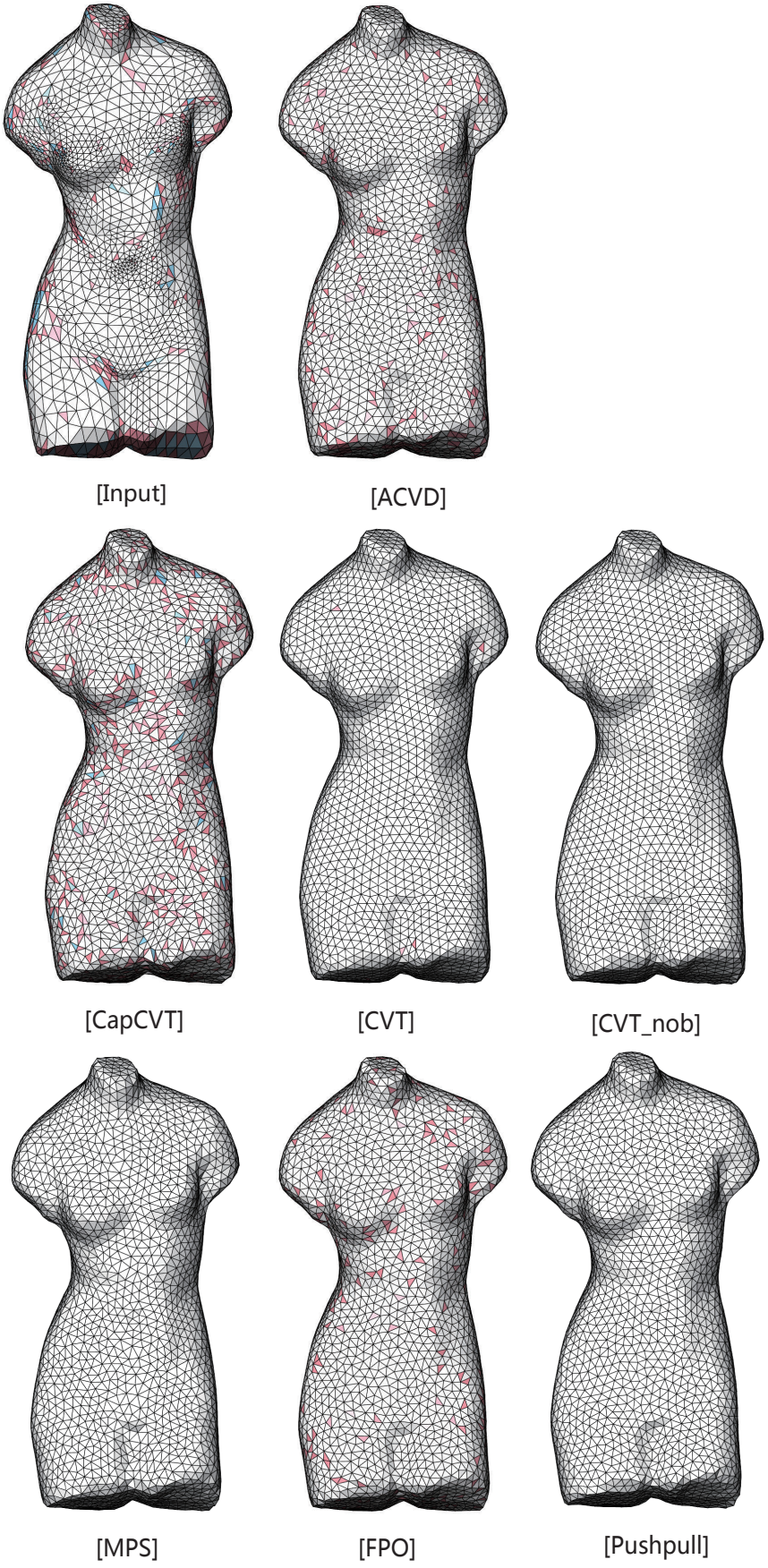


Fig. 4. Uniform remeshing results of the Venus model. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.

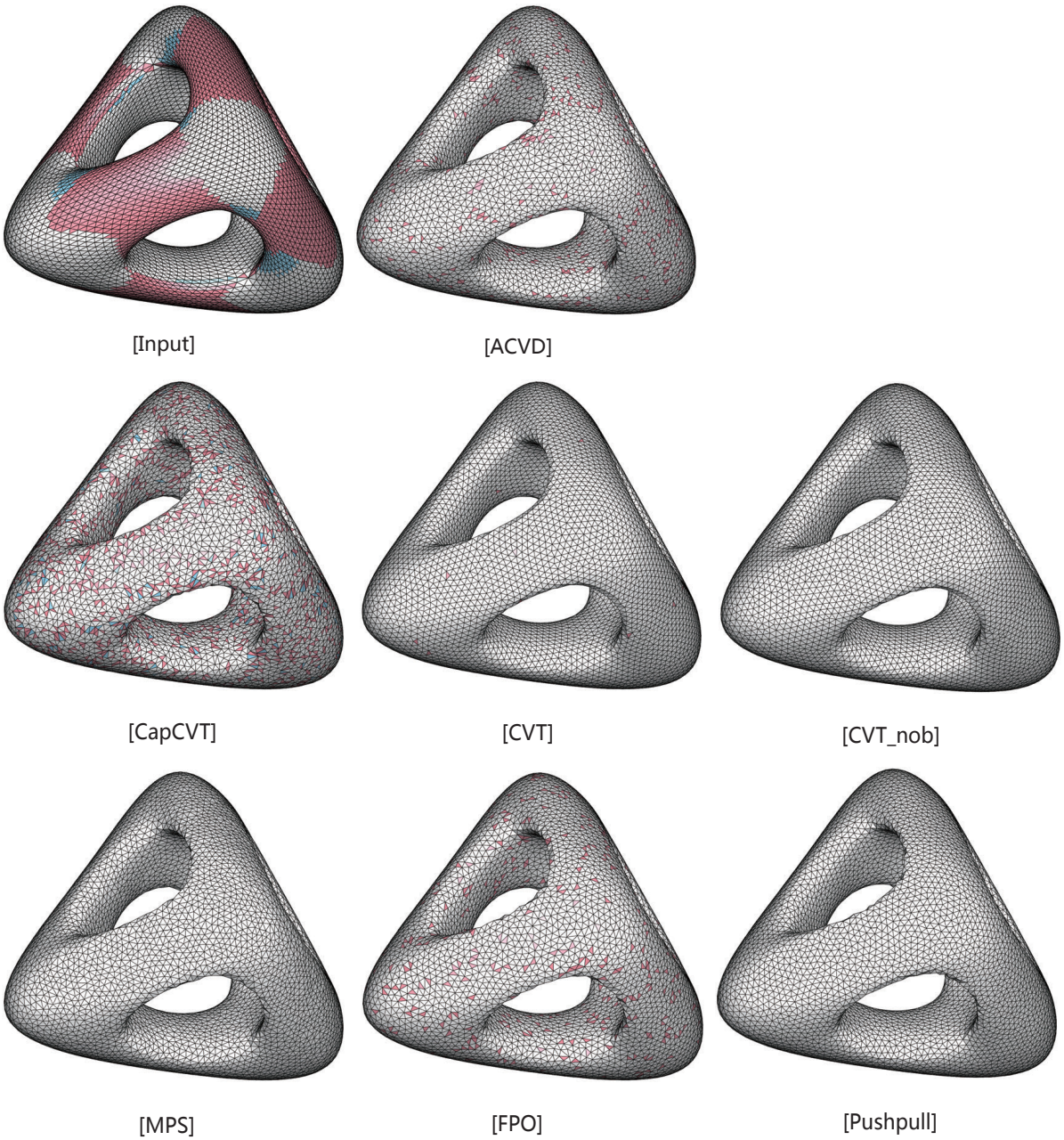


Fig. 5. Uniform remeshing results of the Genus model. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.

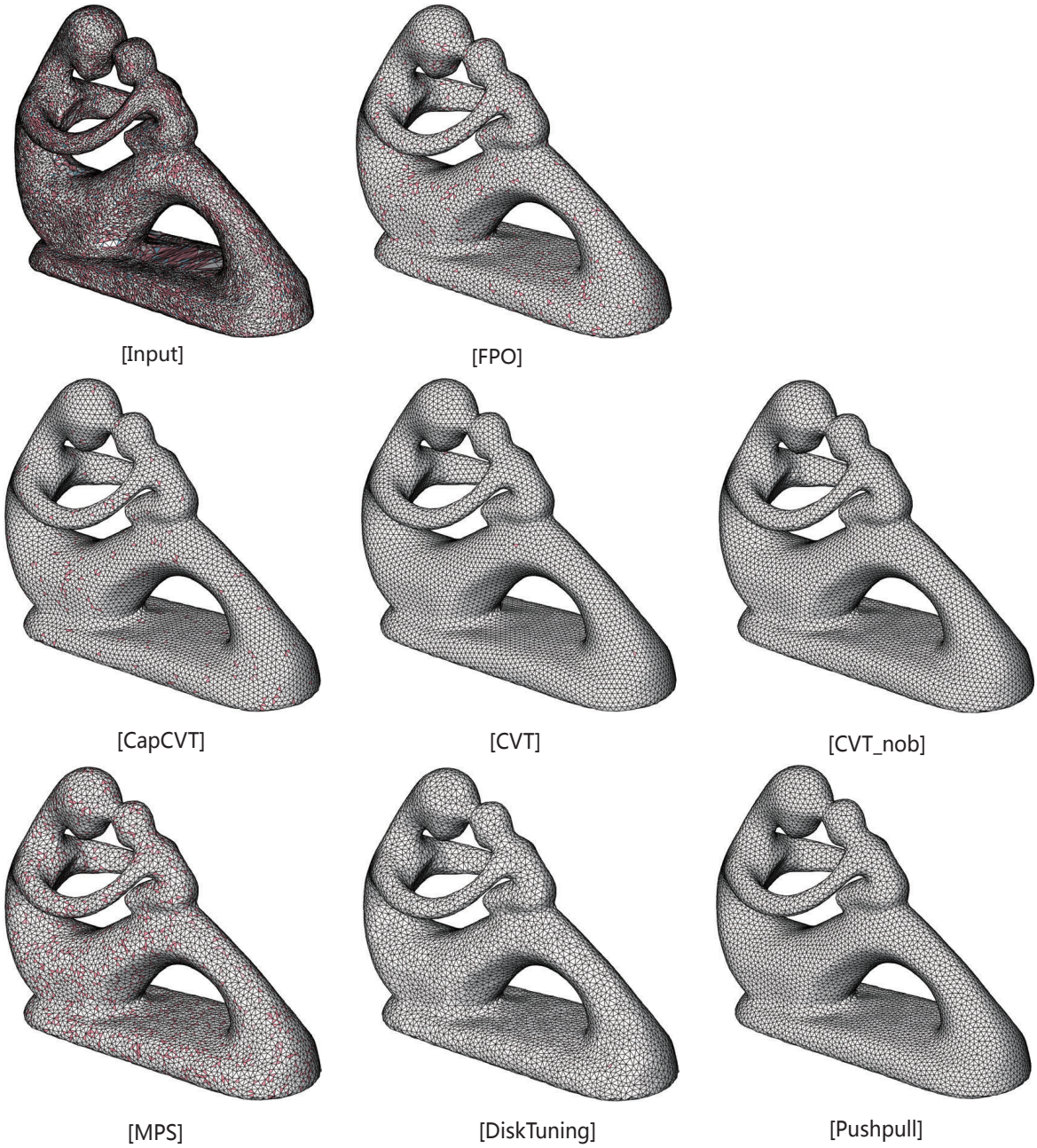


Fig. 6. Uniform remeshing results of the Fertility model. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.

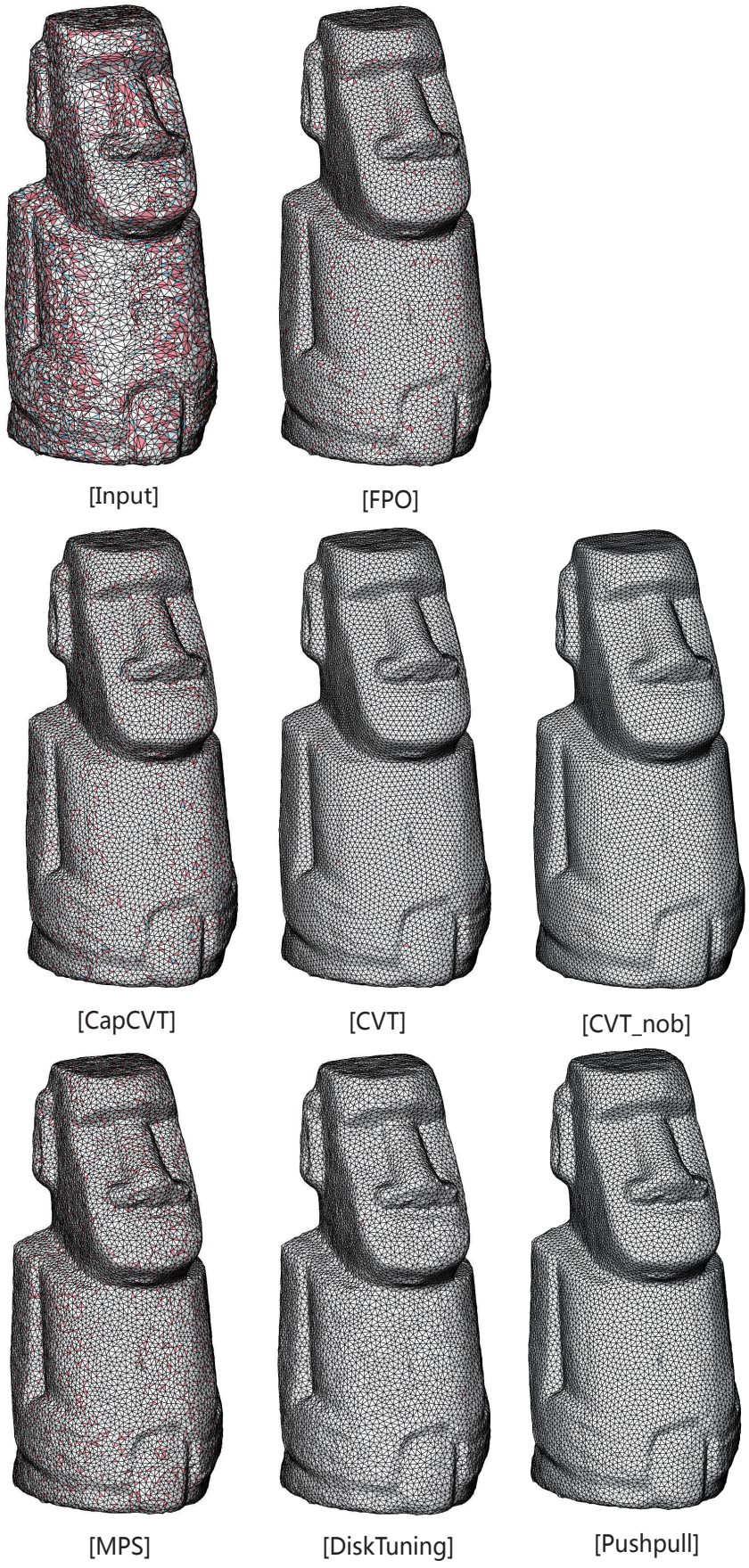


Fig. 7. Uniform remeshing results of the Moai model. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.

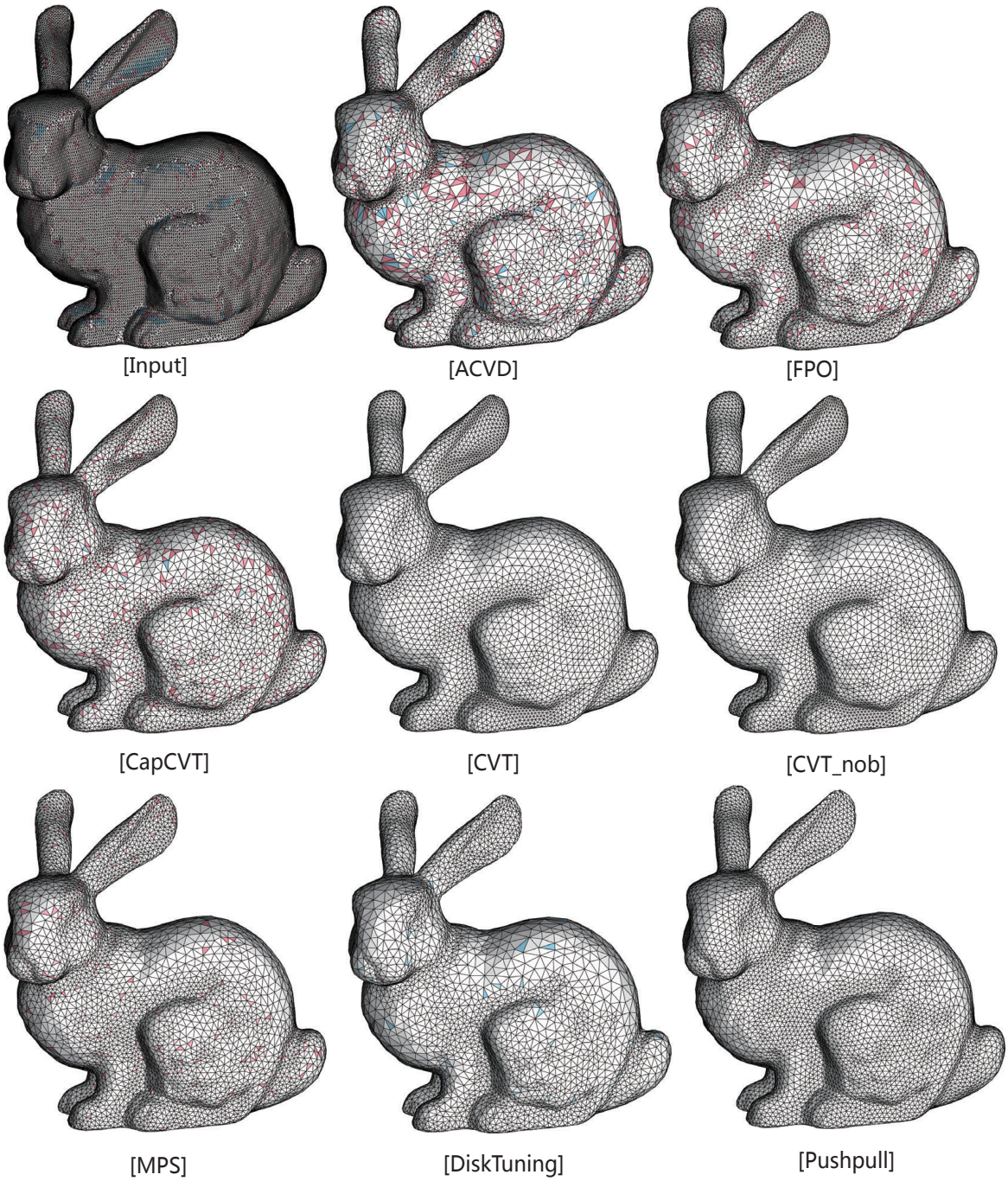


Fig. 8. Adaptive remeshing results of the Bunny model. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.

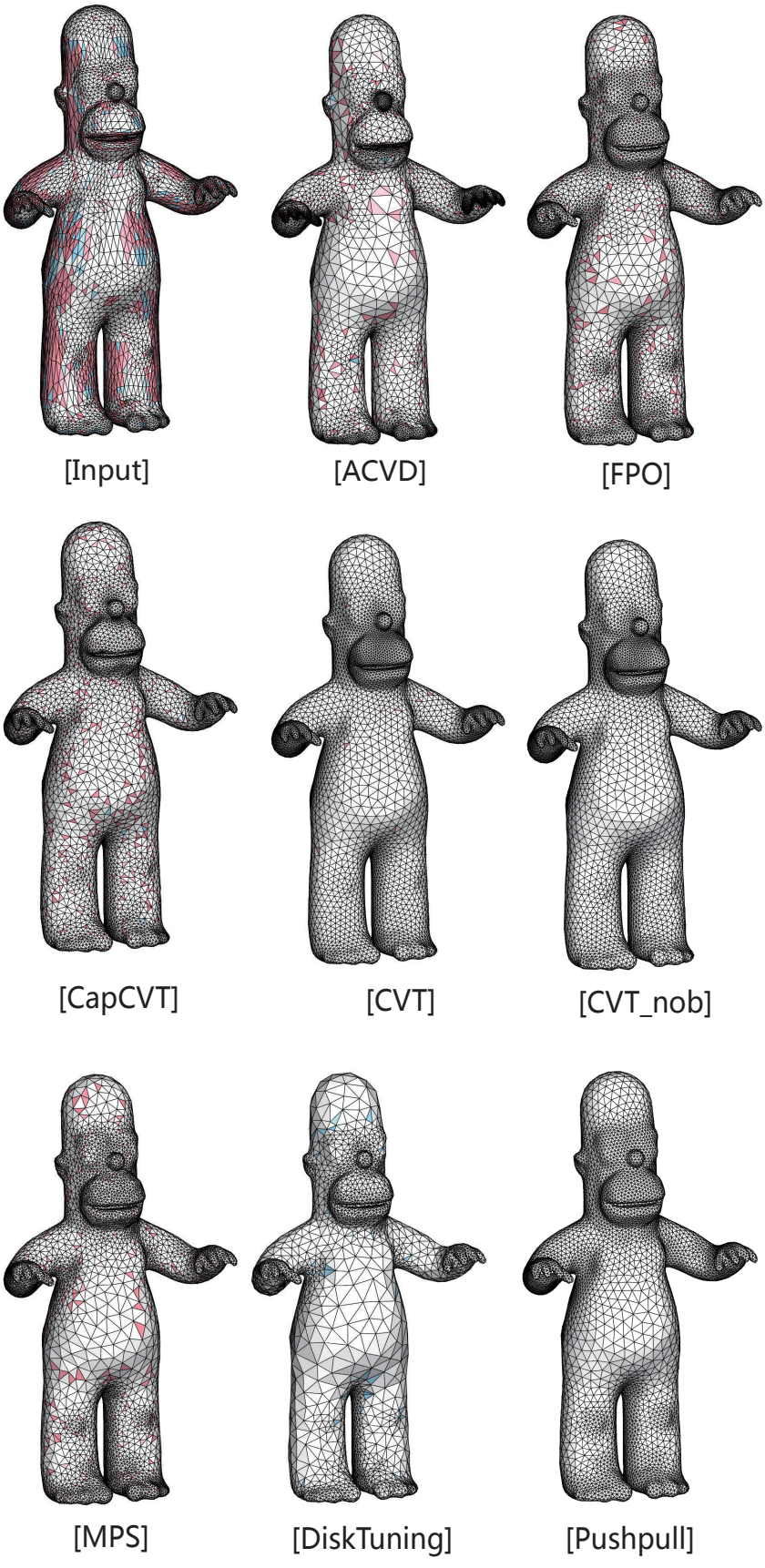


Fig. 9. Adaptive remeshing results of the Homer model. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.

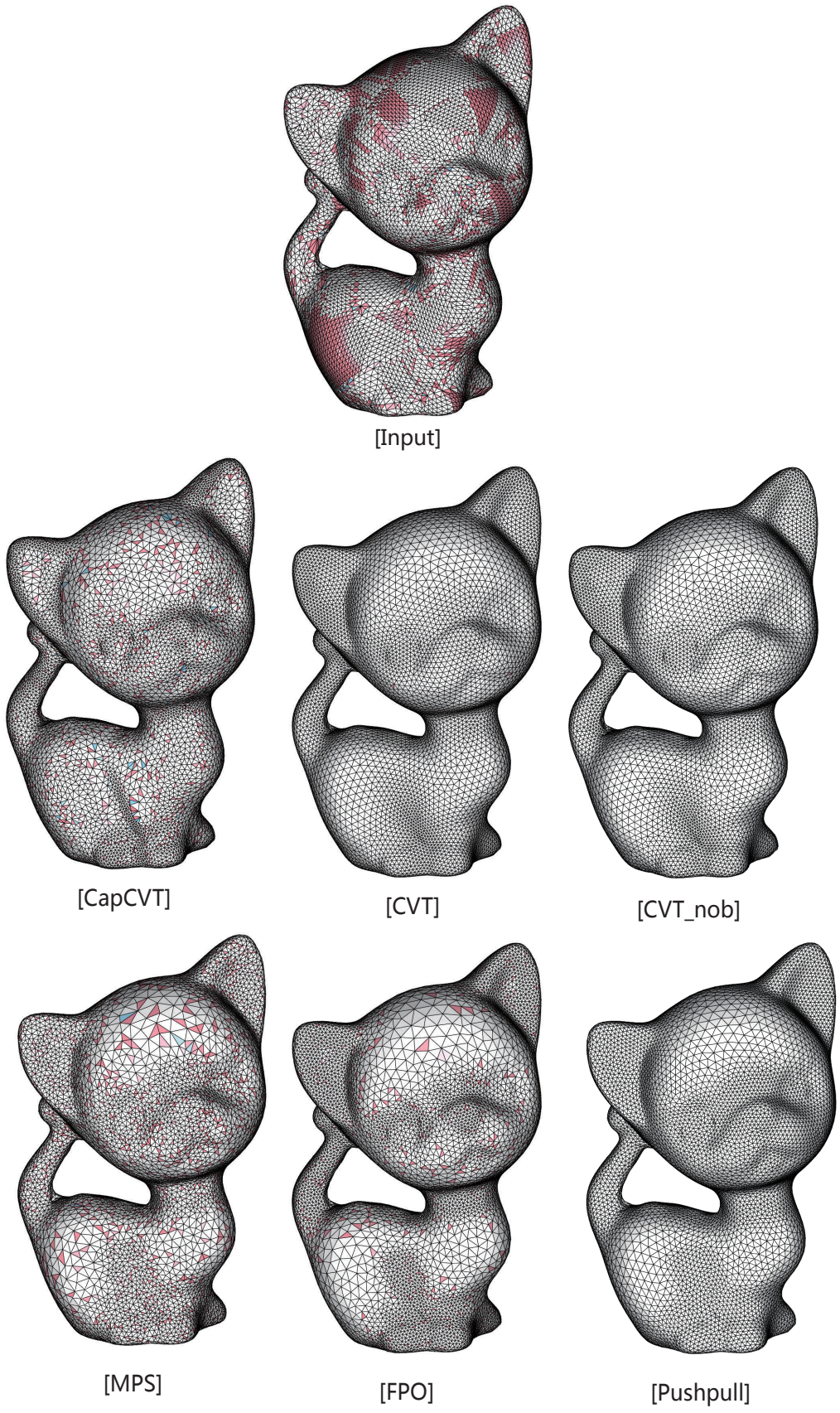


Fig. 10. Adaptive remeshing results of the Kitten model. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.