Moving Level-of-Detail Surfaces – Additional Material

CORENTIN MERCIER, LTCI, Télécom Paris, Institut Polytechnique de Paris, France
THIBAULT LESCOAT, LTCI, Télécom Paris, Institut Polytechnique de Paris, France
PIERRE ROUSSILLON, LTCI, Télécom Paris, Institut Polytechnique de Paris, France
TAMY BOUBEKEUR, Adobe Research, France
JEAN-MARC THIERY, Adobe Research, France

CCS Concepts: • **Computing methodologies** → **Point-based models**.

Additional Key Words and Phrases: Point-based modeling, MLS projections, point set surfaces

ACM Reference Format:
Corentin Mercier, Thibault Lescoat, Pierre Roussillon, Tamy Boubekeur, and Jean-Marc Thiery. 2022. Moving Level-of-Detail Surfaces – Additional Material. *ACM Trans. Graph.* 1, 1 (May 2022), 4 pages. https://doi.org/10.1145/8888888.7777777

1 EXTENDED COMPARISONS

1.1 Comparing quality

In the main paper, we showed comparisons with global APSS (taking the whole input point set when fitting the surface) and APSS using k-nearest neighbors. We add here a comparison of 20 nearest neighbors, 200 nearest neighbors, global APSS, and our approach in Fig. 1. It is clear in this figure that increasing the number of neighbors will not be enough to reconstruct a huge missing part like here the back of the head, whereas the global approach as well as our approach are able to fill-in the hole, albeit with global APSS being much slower. In Figures 2 & 3, we also include comparisons to other methods: Screened Poisson Reconstruction [Kazhdan and Hoppe 2013], Point2Mesh [Hanocka et al. 2020], MPU [Ohtake et al. 2003], MPU-APSS [Xiao 2011] and VIPSS [Huang et al. 2019].

Overall, Screened Poisson Reconstruction [Kazhdan and Hoppe 2013] yields faithful surfaces from the input point set, the reconstruction quality is very good, sometimes better than APSS. However, it cannot be used for filtering : the parameter best suited for such a goal is the reconstruction depth, which does not allow continuous filtering (see paper). We use the implementation of the authors with the Dirichlet boundary type. We vary the depth for the filtering figure of the article.

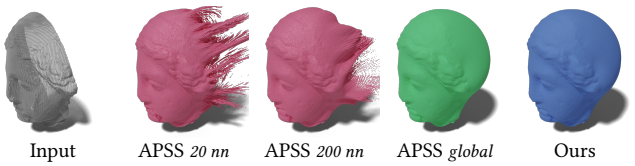

Fig. 1. Comparison with APSS with different number of neighbors.

Often, using MPU [Ohtake et al. 2003] results in holes or outliers in the output, whereas our method does not suffer from such problems. This is especially visible in Fig 2 for the models of the children and the one under it. We used the implementation of the authors with the following parameters : 6 minimum samples (recommended value), quality of fit (fit_epsilon) of 0.005 (recommended value to retrieve details), radius of the sphere of 1.0 (covering parameter, typical value) and a grid resolution of 256 for the ray marching grid. We vary the quality of fit parameter for the filtering figure of the article.

MPU-APSS [Xiao 2011], while being similar to our method, does not allow to "project" the surface into holes for shape completion, as can be seen in Fig. 3 for the polar bear model. We solve this issue for MLoDS with the adaptive octree that we build (see main article). Moreover, MPU-APSS loses the multi-scale view of the shape compared to MPU as only one kernel is used. As the author do not provide an implementation, we reimplemented the approach based on the method described in the paper. We use the same kernel than in our approach with an epsilon parameter of 0.0002 as the precision criterion.

VIPSS [Huang et al. 2019] does not run on the large point sets that we use for our comparisons, so we down-sample the input (to 7000 points) to be able to run this method. This is not the intended case from the authors, which probably explains why the output is not satisfying. Indeed, they do not use the normals and compute them in a first step. We used the authors' implementation with the following parameters : lambda value of 0 (a few others were tested with worst results) and a 100 voxels wide grid for the implicit surfacing (recommended value, increasing it did not improve the results).

Point2Mesh [Hanocka et al. 2020] was run using 25000 iterations, regardless of the point set. The resulting meshes are not smooth, and they feature multiple problems such as triangle inversion or invalid surfaces (particularly visible in meshes with natural "holes", such as the suitcase in Fig.2). The output would probably be better with more iterations, but it would have taken too much time, as the 25000 iterations for every model already took several days to finish. The implementation we used was the one from the authors.

Authors' addresses: Corentin MercierLTCI, Télécom Paris, Institut Polytechnique de Paris, France, corentin.mercier@grosmi.net; Thibault LescoatLTCI, Télécom Paris, Institut Polytechnique de Paris, France, thibault@lescoat.fr; Pierre RoussillonLTCI, Télécom Paris, Institut Polytechnique de Paris, France, roussillon.pierre@gmail.com; Tamy BoubekeurAdobe Research, France, boubek@adobe.com; Jean-Marc ThieryAdobe Research, France, jthiery@adobe.com.

0730-0301/2022/5-ART $15.00
https://doi.org/10.1145/8888888.7777777

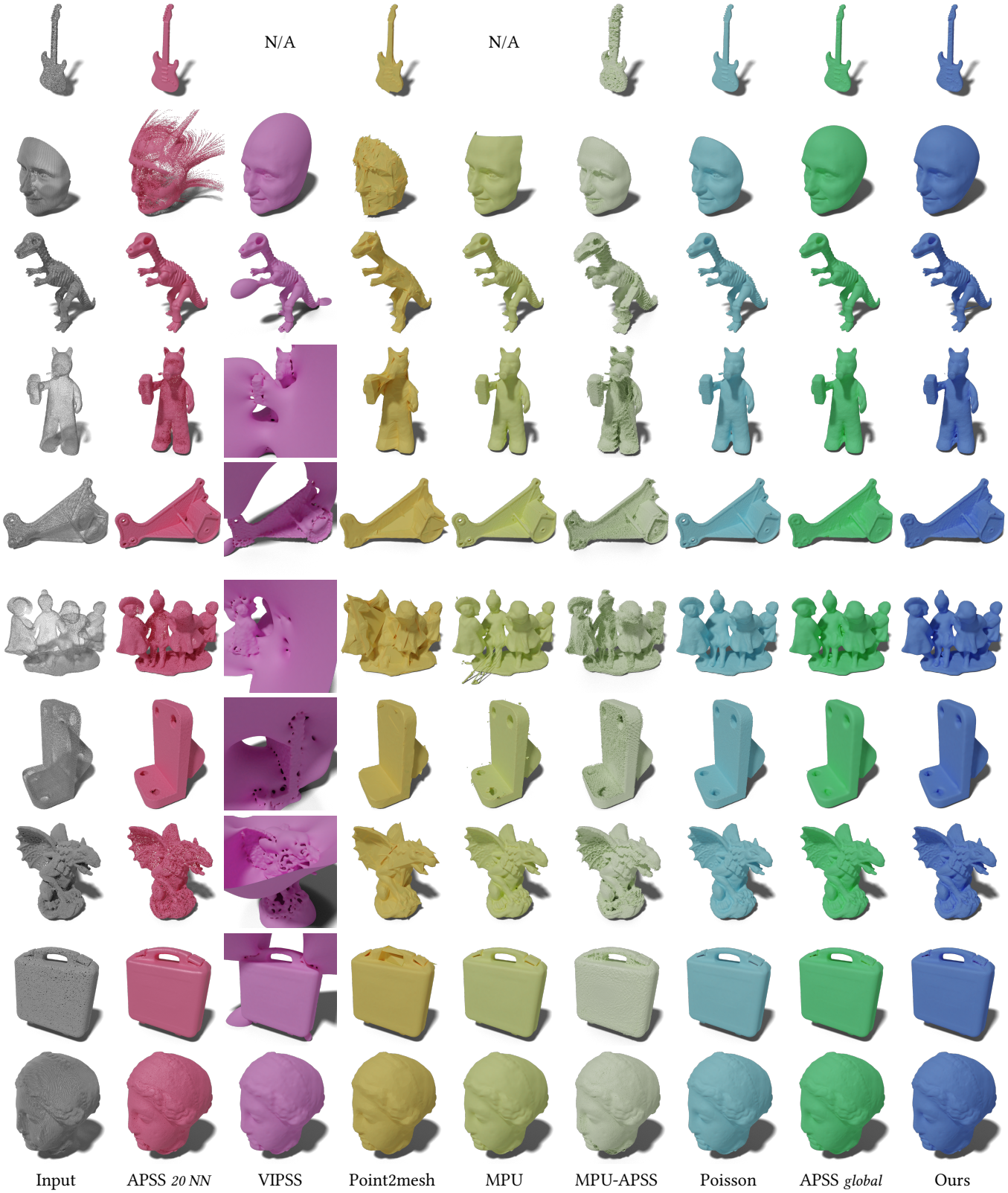

Fig. 2. Comparison with different methods. N/A is displayed when we were unable to obtain a result.

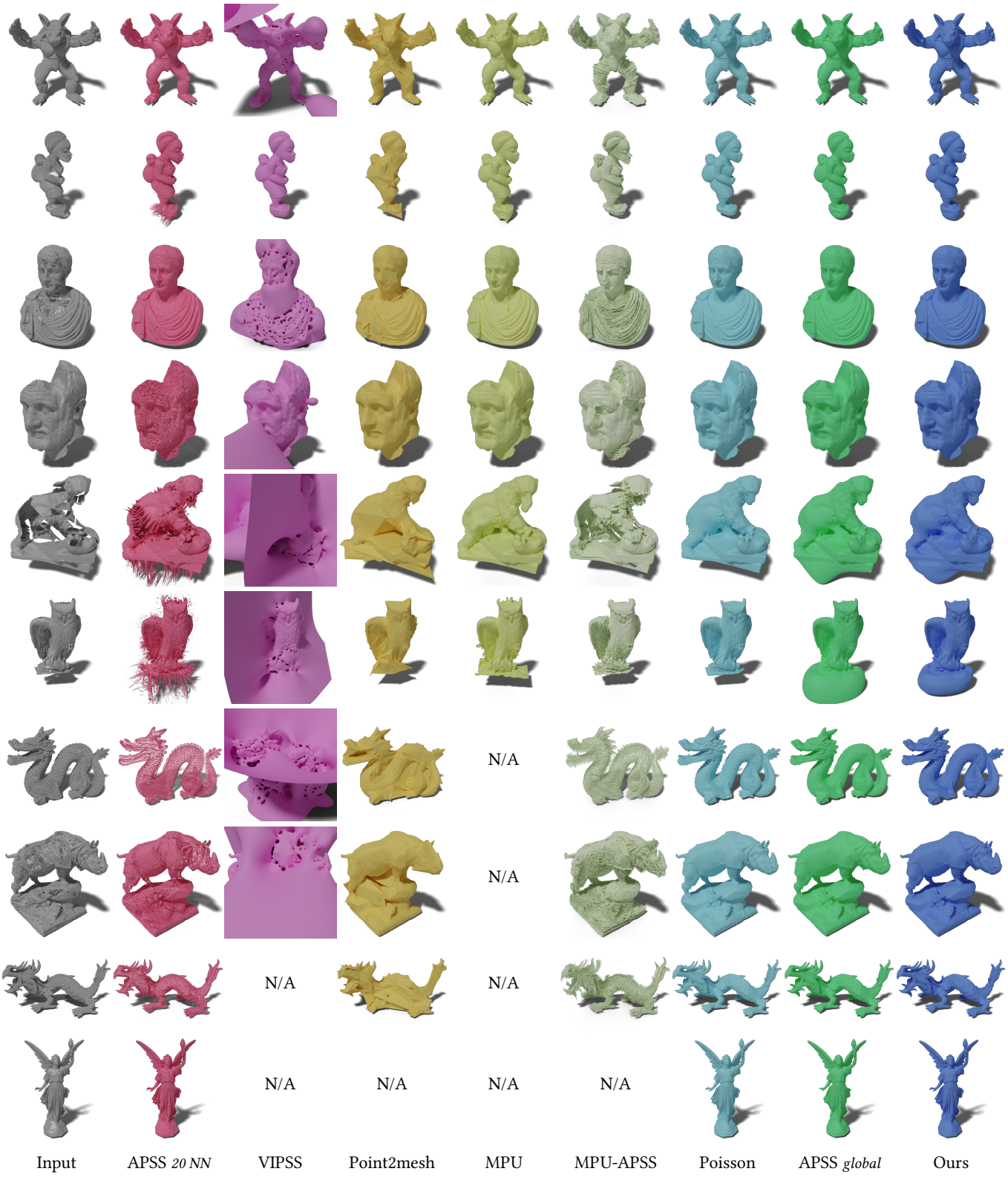

Fig. 3. Comparison with different methods. N/A is displayed when we were unable to obtain a result.

Model	Points	CPU time (μs)					GPU time (μs)	
		20 NN	100 NN	MPU-APSS	MLoDS	GlobalAPSS	MLoDS	Global APSS
Guitar	10007	1.44 ± 0.07	6.67 ± 0.12	0.66 ± 0.05	32.24 ± 1.69	614.69 ± 24.45	2.91 ± 0.17	8.22 ± 0.19
Face	40881	2.78 ± 0.28	5.79 ± 0.42	0.43 ± 0.05	5.82 ± 0.31	851.87 ± 9.91	0.48 ± 0.02	25.25 ± 0.24
Dinosaur	56195	1.45 ± 0.10	6.82 ± 0.16	0.46 ± 0.04	28.62 ± 2.01	3049.67 ± 16.41	2.76 ± 0.16	50.04 ± 0.23
Lord quasimodo	57264	1.43 ± 1.05	5.55 ± 1.21	3.37 ± 0.13	18.13 ± 0.84	2259.44 ± 12.89	1.84 ± 0.07	41.23 ± 0.40
Daratech	61460	3.51 ± 0.08	16.28 ± 0.12	2.89 ± 0.09	71.19 ± 8.56	8719.35 ± 60.22	6.94 ± 0.69	113.17 ± 1.58
Dancing children	71265	1.12 ± 0.53	4.48 ± 0.64	12.05 ± 1.63	13.32 ± 0.61	2334.96 ± 28.60	1.40 ± 0.05	47.35 ± 0.41
Anchor	85127	0.66 ± 0.06	2.77 ± 0.14	15.43 ± 0.23	7.47 ± 0.50	1332.92 ± 52.77	0.85 ± 0.05	51.63 ± 0.42
Gargoyle	95435	1.11 ± 0.36	4.56 ± 0.52	18.75 ± 0.35	12.68 ± 0.48	3000.82 ± 1.72	1.40 ± 0.05	63.58 ± 0.48
Suitcase	99886	0.91 ± 0.32	3.43 ± 0.50	3.76 ± 0.17	9.93 ± 0.63	1958.61 ± 13.12	1.05 ± 0.05	64.22 ± 0.38
Igea	134346	1.07 ± 0.28	4.47 ± 0.40	1.17 ± 0.08	12.98 ± 0.72	3978.08 ± 19.56	1.39 ± 0.06	93.70 ± 0.57
Armadillo	172974	2.26 ± 0.12	10.97 ± 0.16	0.91 ± 0.06	39.43 ± 2.83	15971.80 ± 41.18	3.83 ± 0.18	220.64 ± 1.23
African Statue	220317	2.05 ± 0.39	8.28 ± 0.40	0.85 ± 0.09	32.07 ± 2.34	11956.50 ± 21.60	3.03 ± 0.19	200.38 ± 1.03
Napoleon	495000	1.29 ± 0.20	5.00 ± 0.24	5.74 ± 0.19	15.20 ± 1.34	12165.80 ± 23.29	1.64 ± 0.14	344.54 ± 2.02
Bearded Man	499500	1.04 ± 0.26	3.60 ± 0.36	0.97 ± 0.16	10.07 ± 1.01	9223.72 ± 18.41	1.28 ± 0.14	320.86 ± 3.53
Eisbar	781865	5.75 ± 1.09	14.62 ± 0.46	13.69 ± 0.25	24.56 ± 1.89	44060.40 ± 154.45	2.16 ± 0.16	696.79 ± 7.69
Owl	1031960	5.40 ± 4.55	13.15 ± 4.67	1.96 ± 0.17	30.90 ± 2.42	67452.00 ± 168.53	2.46 ± 0.18	1076.94 ± 23.67
Dragon	1180060	2.04 ± 1.90	5.60 ± 2.40	27.46 ± 0.27	37.32 ± 2.80	26116.10 ± 85.07	4.26 ± 0.35	776.88 ± 5.51
Rhinoceros	1410356	6.28 ± 6.60	18.05 ± 6.75	45.51 ± 0.51	32.87 ± 2.79	86970.40 ± 137.95	3.39 ± 0.22	1418.64 ± 10.01
Xyzrgb Dragon	3609455	5.06 ± 10.66	7.21 ± 11.82	108.28 ± 0.94	8.65 ± 1.90	17003.30 ± 642.83	1.77 ± 0.57	2078.57 ± 25.93
Lucy	14027872	10.94 ± 4.35	36.19 ± 3.92	N/A	146.31 ± 32.32	1850291.25 ± 1284.64	16.24 ± 0.85	45532.1 ± 4580.6

Table 1. Average time to project one point (mean + standard deviation). The MPU-APSS timings are obtained using our implementation of the method. Lucy was unable to finish for memory reasons.

The N/A present in Fig. 2 and Fig. 3 correspond to models for which the given method was unable to finish, either for memory reasons or because the code crashed.

1.2 Comparing computation time

We used a GTX 1080Ti and a Xeon E5-1650 v4 (3.6GHz, 12 threads) for all our tests.

First, we compare timings for APSS-related methods in Table 1, namely the time to project one point onto the surface. As seen in the main paper, our method approaches the speed of using a few neighbors but with the same quality as using the full input point-set. The latter (Global APSS) is multiple orders of magnitude slower, making is ill-suited for moderately large inputs (see Fig. 4). MPU-APSS [Xiao 2011] can often be even faster than using 20 nearest neighbors, albeit with a notable cost in quality. Our GPU implementation reduces by an order of magnitude the computation time to project one query point, compared with the CPU implementation.

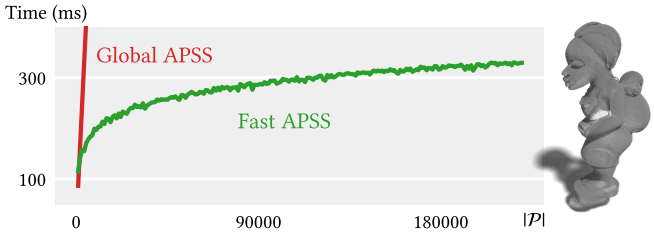

Fig. 4. While the time to project 100000 points is in $O(|\mathcal{P}|)$ for GlobalAPSS, it is in $O(\log(|\mathcal{P}|))$ for FastAPSS.

Although we do not have precise timings for other methods compared in these additional materials, we observed that Screened Poisson Reconstruction [Kazhdan and Hoppe 2013] is in general very fast to determine a surface from the input point-set. MPU [Ohtake et al. 2003] is also quite fast in general.

The slowest methods were VIPSS [Huang et al. 2019] – which runs in $O(N^3)$, with N the size of the input – and Point2Mesh [Hanocka et al. 2020] – which required 25000 iterations. For numerous test models, these methods took multiple days to complete.

REFERENCES

Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: A Self-Prior for Deformable Meshes. *ACM Transactions on Graphics (TOG)* 39, 4, Article 126 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392415

Zhiyang Huang, Nathan Carr, and Tao Ju. 2019. Variational Implicit Point Set Surfaces. *Trans. Graph.* 38, 4 (july 2019), 124:1–124:13. https://doi.org/10.1145/3306346.3322994

Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 29.

Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. 2003. *Multi-level partition of unity implicits.* Vol. 22. ACM.

Chun-Xia Xiao. 2011. Multi-Level Partition of Unity Algebraic Point Set Surfaces. *J. Comput. Sci. Technol.* 26, 2 (March 2011), 229–238.