Supplementary for ROAR: Robust Adaptive Reconstruction of Shapes Using Planar Projections

1 RECONSTRUCTION IN-THE-WILD IMPLEMENTATION DETAILS

Throughout our experiments, we use the same hyper parameters (Section 1.2). We used the coefficients $\lambda_1 = 3$ and $\lambda_2 = 1$ (Eq. (2)) for all shapenet experiments. Our learning rate was set to $5e^{-2}$ and we iterated using Move Vertices (sec. 3.1) for 200 steps. We ran the Face Collapse block for each iteration, and the Face Split Block each 10 iterations.

1.1 Metrics Calculation

We describe here in detail how the metrics in Table 1 in the main paper were computed. Chamfer L_2: we randomly sample 200k samples from the surface of the target proportionally to face areas and of the result, and compute the average bi-directional Euclidian distance between nearest neighbors. Cosine Similarity: exactly as Chamfer L_2, only we compute the dot product between the nearest neighbors' normals. Triangle Quality: we compute the median aspect ratio of the faces per mesh (and average upon that). Aspect ratio is defined as the ratio between circumcircle radius over twice the inner radius of the triangle. Self Intersections: we count the number of faces intersecting with others and divide by the total amount of faces. We deducted from the calculation any face that intersects with another which can be trivially resolved by adding epsilon to its vertices locations in the direction of their normals. These type of self intersections are common when the target exhibits large flat thin areas. Non Manifold Edges: average number of edges which have more than 2 incident faces. Non Manifold Vertices: average number of vertices where the link is not a single loop of edges [13].

1.2 Additional Hyper Parameters

An exhaustive list of thresholds and hyper parameters used for ROAR are detailed below. We fixed them for all experiments except when explicitly mentioned otherwise (for ablations).

- Image resolution - we used a $512x512$ resolution for all renderings of the objects.
- Face removal threshold for initialization - after rendering 36 views of the target, faces that have a pixel count of less than 80 times their twin are removed. This faces are duplicated faces that face inwards, and hence can be filtered out.
- Number of sampling points - we use up to 200k points for sampling the target mesh in the Mesh Preprocessing step.
- Face split amount, or supersamples per face - we set the number of super samples per face to be proportional to its area divided by 0.2% of the diagonal size of the bounding box of the mesh (defined as delta in the original paper [3]). We do not allow the amount to go beyond a value of 420.
- Face split score threshold - for face splitting, we filter out candidates to split who don't meet a minimum score of 0.02
- Tangential smoothing threshold - when performing tangential smoothing, we do not smooth vertices which have their absolute normal size smaller than 0.98. This avoids moving vertices on very sharp spikes or edges of a shape, facilitating the triangulation to align to crease angles.
- Number of neighbors for projection loss support - we used 10 neighbors to compute the projection loss.
- Collapse Face threshold - we increased the slack of the self-intersection estimator $n_v \cdot n_f < 0$ by adding $\delta = 0.6$ to the RHS, which allows more faces to be considered for collapse.
- Collapse Face normal threshold - we set a threshold that removes candidates from being collapsed when their absolute normal size is smaller than 0.98 (similar reason to the tangential smoothing threshold).
- Number of viewpoints - We used 36 different views for rendering the target and source mesh used to compute the render loss.
- Dihedral angle threshold for edge flips - we set a threshold to prevent edges with a small dihedral angle (0.95) from flipping during this step, again due to the same sharp feature reasoning.
- Collapse Face quality threshold - before collapsing an edge of a face, we check if the resulting triangles quality is worse than 5, and remove candidates that don't meet this threshold. We also check if the normals of the result are flipped and remove candidates for which this condition is met.
- Beam angle horizon - half of the beam sweep angle. Defined relative to a given central vector (10°).
- Number of beam rays in azimutal direction - number of rays in the azimuth direction (4).
- Number of beam rays in elevation direciton - number of of rays in the azimuth direction (3).

1.3 Hyper Parameters for Other Methods

Parameters used to apply other methods were selected by trial and error to yield good performance, and are described below. For Manifold and Manifold+ [10; 11], all default parameters were used. For Continuous Remeshing [15], we converted the original scheme to use flat shade rendering and benefit from higher quality gradients when dealing with triangle soups. For TetWild and fTetWild [8**?**], we used all default parameters, but extract the boundary surface from the result. For [14] we used default parameters. For [7] we used 6000 iterations and pooling layers of 0.1, 0.0, 0.0, 0.0, which are the parameters used for their Bull example. For Screened Poisson [12], we used the implementation provided by MeshLab [5], with a depth of 8. For Alpha Wrapping [2], we used an alpha value of 300 and offset of 1000. For any instance of QSLIM [6] used as decimation, we used the MeshLab [5] implementation with default parameters, except the "preserve topology" and "preserve normals" options which were turned on.

1.4 Face Split Further Detail

As mentioned, we project using an approximate sampling scheme. Its performance relative to the full point-to-mesh projection can be seen in Figure 1.

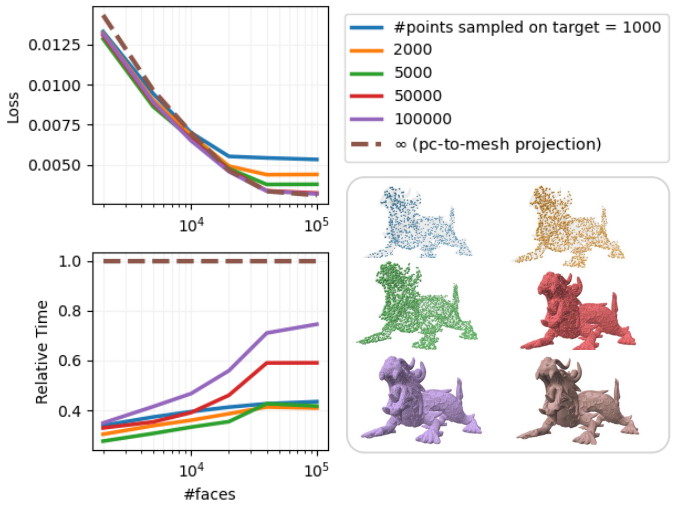

Fig. 1. Planar Projection Approximation: we sample points randomly on the surface of the target mesh, and approximate pointcloud-to-mesh projection using nearest neighbors. for 50k samples, we converge to the pointcloud-to-mesh projection for face budgets of up to 100k, with a significant speedup

.

2 IMPROVED SAMPLING OF TRIANGLE SOUP FEATURES

Since both face splitting and the planar projection loss approximate a true projection using k-NN sampling, we benefit from having a denser sampling at sharp edges. The problem is that these edges can not be detected on triangle soups, due to noisy topology. We overcome this by sampling sharp edges on the initial mesh generated by MANIFOLD [10] and projecting them back to the target mesh. The samples are drawn randomly from all the edges for which the absolute value of dot product of the incident face normals is lower than 0.5. This procedure captures the "outline" of the triangle soup (2). We added 30k such points to our cleaned point cloud.

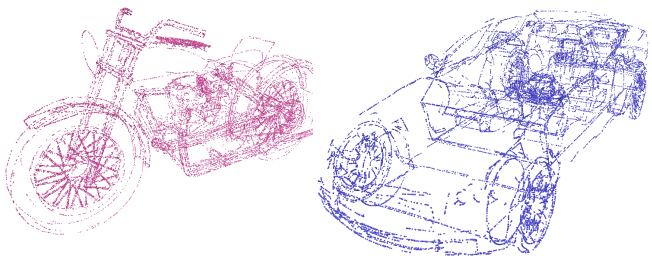

Fig. 2. Some sampling results using our procedure to sample sharp edges on the triangle soups

.

2.1 Failures

Table 1 sums up the percentage of meshes that weren't used for metric calculations and the average time it took to process each mesh for all methods. Note that for Ours, Manifold and Manifold+, "failures" consisted of the post process decimation algorithm implementation [5] failing to decimate to the proper face budget with settings that do not permit change in topology. In terms of the methods themselves there were 0 failures. For Tet-Wild, failures are due to being unable to converge and for Point2Mesh, exceeding a reasonable reconstruction time of 30 minutes.

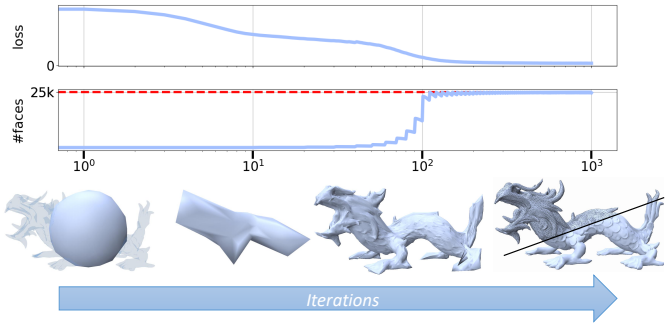

Fig. 3. Time lapse. Given a certain face budget (25k) and starting from a subdivided sphere, we observed that reaching the target face budget is not a good criterion for termination, as the loss continues to decrease due to topology refinements

.

Table 1. Failure Percent and Average Running Times for comparisons over 550 meshes (10 per class) in ShapenetCoreV2.

Method	Fail % Face Budget 500/1k/5k/10k	Avg. Running Time (s)
Ours	10/6.2/0.5/0	120
Manifold [10]	5.6/2.7/0.18/0	1.2
Manifold+ [11]	8.0/4.4/0/0	4.5
Continuous Remeshing [15]	0	60
Tet-Wild [9]	3.2	271
fTet-Wild [8]	0	90
RIMLS [14]	0	78
Point2Mesh [7]	63*	1625
Screened Poisson [12]	0	11
3D Alpha Wrapping [2]	0	22

* Fails due to forced termination after a 30 minutes timeout was reached.

2.2 Face Budget and Post-Process

We conducted two experiments to evaluate the effects of the face budget limit on results versus performing decimation to the face budget as a post process step (Figure 4). Image loss was significantly lower when performing post process decimation as opposed to directly optimizing towards a face budget (and never passing it). Experiments were performed over 11 high resolution meshes mentioned in Section 4.1. Another important aspect of the face budget is whether reaching it can be used as a good termination criterion for optimization (assuming the initial source mesh has low resolution).

We tested this (see Figure 3) and found that the loss keeps decreasing even after reaching the target budget for a non-negligible amount of iterations. This might suggest further optimization of scheduling the blocks can improve results.

3 NEURAL SDF RECONSTRUCTION IMPLEMENTATION DETAILS

For all the experiments, the same hyper parameters were used to perform reconstruction of neural SDFs: for initialization, we used a simple voxelization scheme using a coarse $64x64x64$ grid and declaring samples inside the shape when $SDF(x) < 0$. We ran ROAR for 700 iterations of optimization with the full reconstruction pipeline turned on, and a face budget of 50k faces. We introduce the SDF projection loss after 500 iterations multiplied by a coefficient of 0.1. All other settings were kept at default (Section 1).

Since the 3D meshes that were used to create the BlendedMVS dataset are given in files that describe only the geometry of the object, and not the stage it is placed upon, and because the neural SDF models did in fact capture the stage, it was necessary to separate them for computation of Chamfer distance from the ground truth. To this end, we defined 2D planes that were placed between the object and the stage for all scans, and sliced both our result and the Marching Cubes results using the planes such that triangles with all vertices below the planes were discarded. We then decimated the marching cubes results until it had 50k triangles for fair comparison and sampled it as described in the main paper. We confirmed with the authors of Yariv et al. [17] that this process was done in their computation of the Chamfer distance as well.

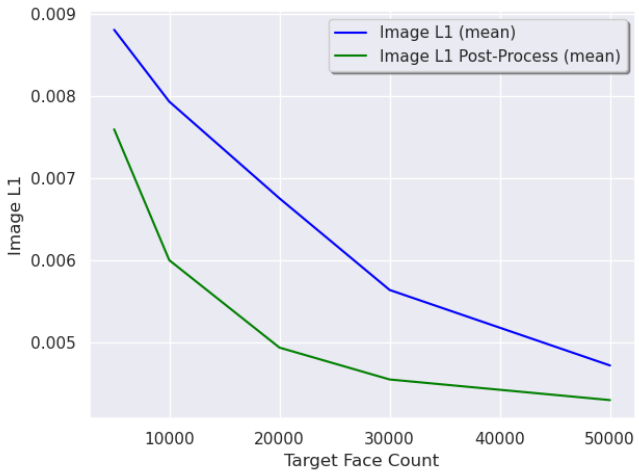

Fig. 4. Face Budget. Given a face budget, we tested the difference between optimizing towards it directly versus optimizing towards a much higher budget and decimating the result a post process step [6]. We observed the more relaxed approach (which allows going beyond the required budget) yields better results.

4 RANGE SCAN RECONSTRUCTION IMPLEMENTATION DETAILS

Results of reconstruction can be seen in Figure 5. Scans were converted into patches (triangular meshes with holes) by connecting nearest neighbors in the scan and triangulating the resulting grid. We eliminate any triangle from the patch for which the angle between the view angle it was captured from and its normal are larger than a threshold to reduce sampling noise. For each object, multiple such patches exist so they were aligned using an ICP variant [5] initialized by the provided transformations. Aligned patches were rendered "as is", with no further adjustments.

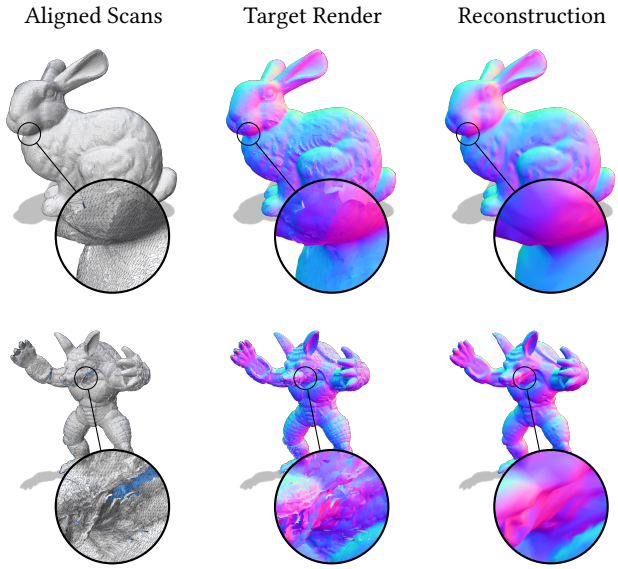

Fig. 5. Range Scans Reconstruction. Despite missing data and render artifacts of the aligned raw scans, we manage to reconstruct a fairly accurate shape from the target renders.

4.1 Datasets Details

We used the [4] dataset for most of the analysis, and 9 scans from [16]. Additionally, we used the following 3D meshes for the ablation study: Bunny, Armadillo, Dragon, Fandisk, Lucy, Nefertiti, Horse, Smilodon [15], Thundercrab (ID:133582)[18], Deer (ID:921798)[18], Engine(ID:669972)[18], Camel (ID:482274)[1].

REFERENCES

[1] 2020. Camel Lowpoly Free 3D model - .OBJ, .STL - open3dmodel. https://open3dmodel.com/3d-models/camel-lowpoly_482274.html
[2] Pierre Alliez, David Cohen-Steiner, Michael Hemmer, Cédric Portaneri, and Mael Rouxel-Labbé. 2023. 3D Alpha Wrapping. In *CGAL User and Reference Manual* (5.5.2 ed.). CGAL Editorial Board. https://doc.cgal.org/5.5.2/Manual/packages.html#PkgAlphaWrap3
[3] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. 2002. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proceedings. IEEE international conference on multimedia and expo*, Vol. 1. IEEE, 705–708.
[4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository.* Technical Report arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago.

[5] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, Vittorio Scarano, Rosario De Chiara, and Ugo Erra (Eds.). The Eurographics Association. https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

[6] Michael Garland and Paul Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* 1997 (07 1997). https://doi.org/10.1145/258734.258849

[7] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: A Self-Prior for Deformable Meshes. *ACM Trans. Graph.* 39, 4, Article 126 (aug 2020), 12 pages. https://doi.org/10.1145/3386569.3392415

[8] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4, Article 117 (July 2020), 18 pages. https://doi.org/10.1145/3386569.3392385

[9] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201353

[10] Jingwei Huang, Hao Su, and Leonidas Guibas. 2018. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698* (2018).

[11] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. 2020. ManifoldPlus: A Robust and Scalable Watertight Manifold Surface Generation Method for Triangle Soups. *arXiv preprint arXiv:2005.11621* (2020).

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

[13] Mathieu Desbrun Keenan Crane, Fernando de Goes and Peter Schröder. 2013. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 courses* (Anaheim, California) *(SIGGRAPH '13)*. ACM, New York, NY, USA, 126 pages.

[14] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. 2009. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 493–501.

[15] Werner Palfinger. 2022. Continuous remeshing for inverse rendering. *Computer Animation and Virtual Worlds* (2022), e2101.

[16] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. 2020. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1790–1799.

[17] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems* 34 (2021), 4805–4815.

[18] Qingnan Zhou and Alec Jacobson. 2016. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016).