# Reconstructing Thin Structures of Manifold Surfaces by Integrating Spatial Curves: Supplementary materials

Shiwei Li Yao Yao Tian Fang Long Quan
{slibc|yyaoag|quan}@cse.ust.hk tianft@gmail.com
The Hong Kong University of Science and Technology

## 1. Overview

This document presents additional results and comparisons that are not put into the main paper due to space limitation. The main content includes:

Section 2: Full qualitative results;
Section 3: EPFL datasets;
Section 4: Breakdown experiment;
Section 5: Evaluation of the negative influences;
Section 6: Comparisons of 3D curve reconstruction;
Section 7: Further explanations of Curve-conformed Delaunay Refinement;
Section 8: Runtime performance.

## 2. Full qualitative results

In this section, we show the qualitative results of all datasets used in our experiments.

### 2.1. Synthetic datasets

We have used six synthetic datasets for quantitative evaluations in the main paper in Section 4.1, namely *chair*, *chair2*, *fence*, *holder*, *lamp* and *pylon*. Here we present qualitative results of them generated by three state-of-the-art methods, our method and ground truth. Three state-of-the-art methods are introduced in the main paper in Section 4.1.

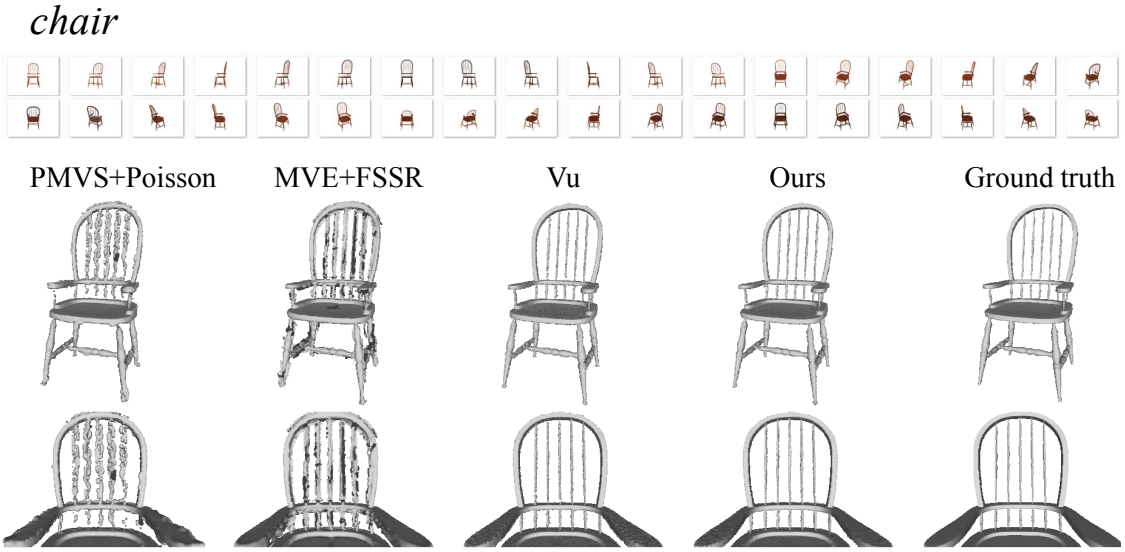


Figure 1: The *chair* dataset. Upper: input images. Lower: mesh reconstructions by four methods and ground truth.

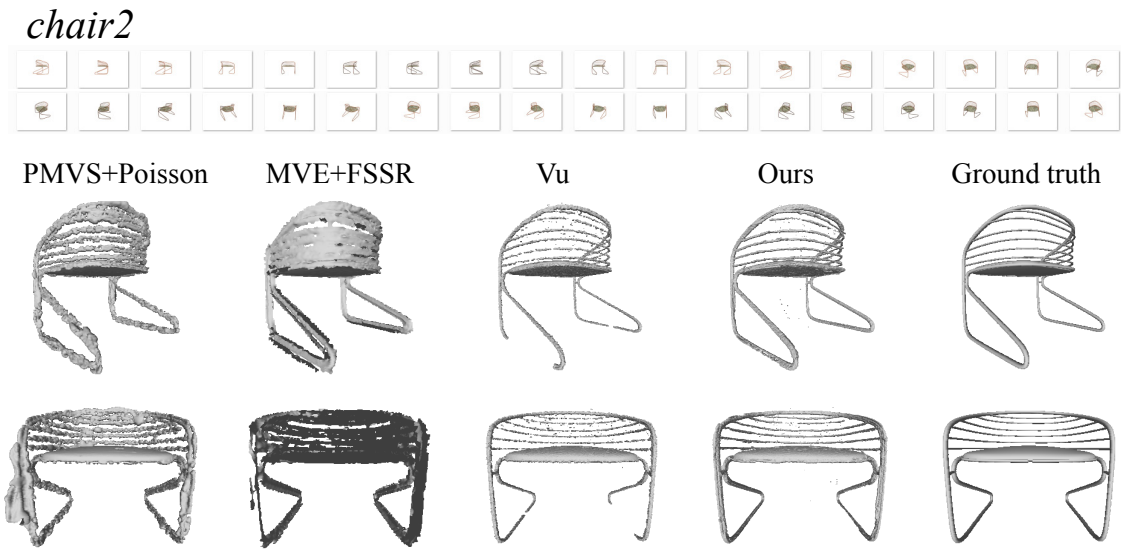


Figure 2: The *chair2* dataset. Upper: input images. Lower: mesh reconstructions by four methods and ground truth.

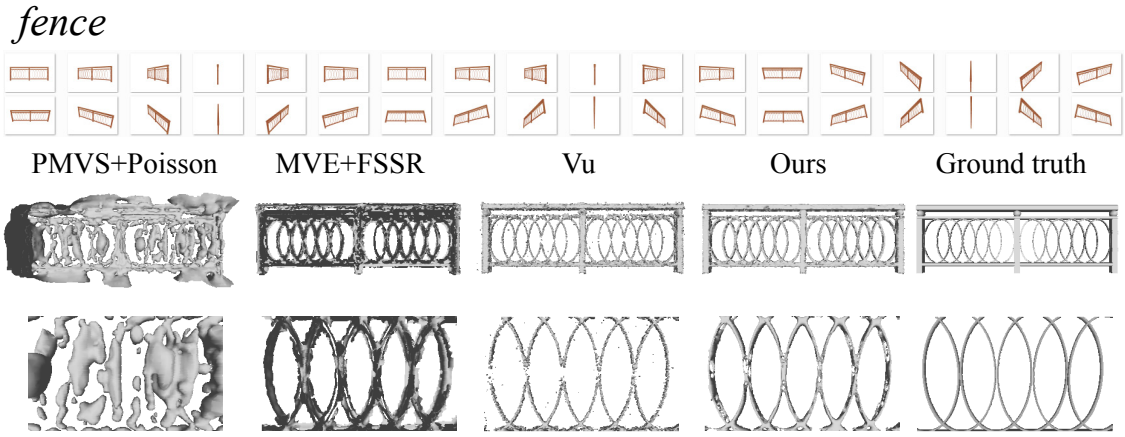


Figure 3: The *fence* dataset. Upper: input images. Lower: mesh reconstructions by four methods and ground truth.

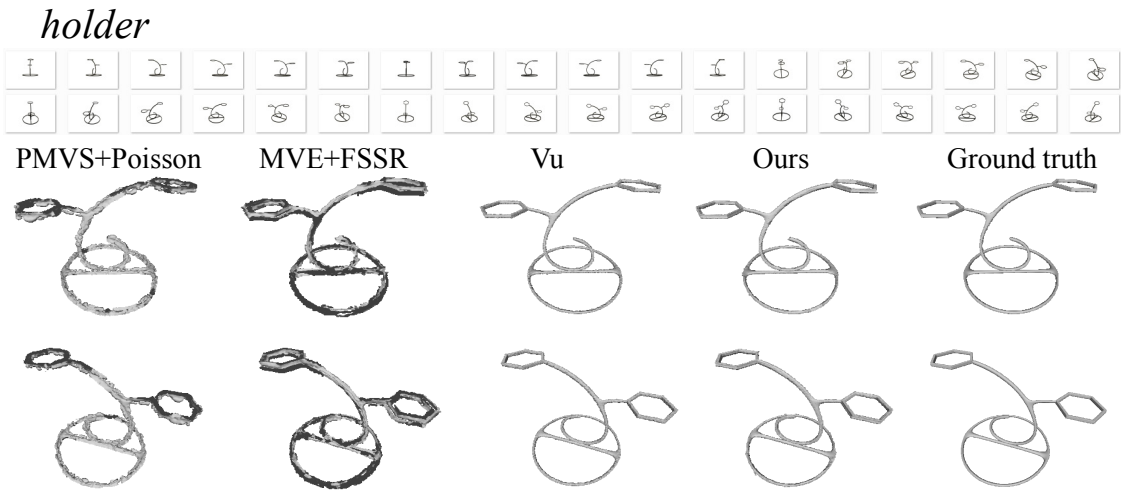


Figure 4: The *holder* dataset. Upper: input images. Lower: mesh reconstructions by four methods and ground truth.

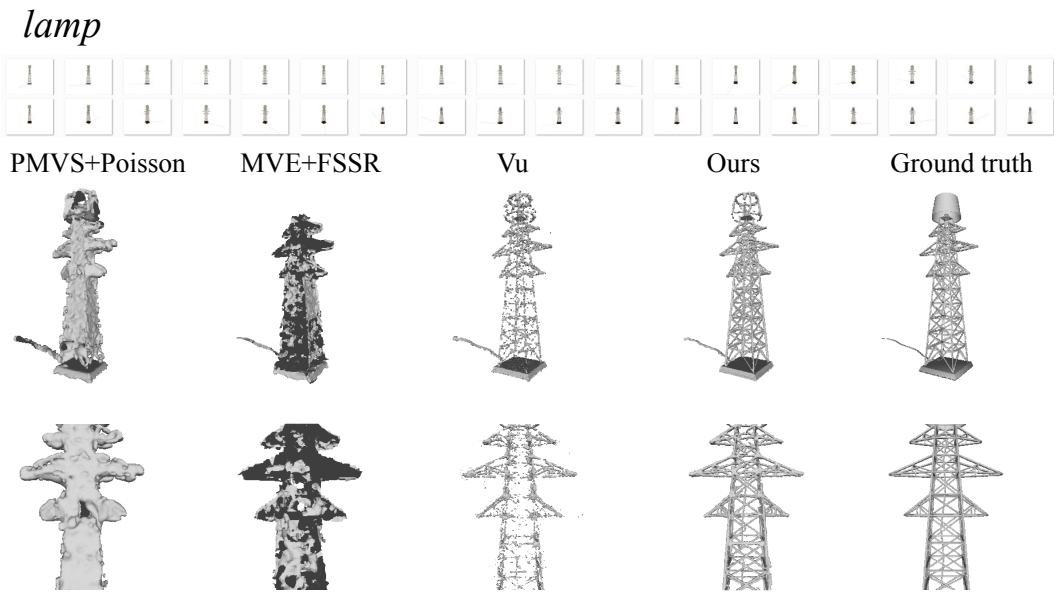


Figure 5: The *lamp* dataset. Upper: input images. Lower: mesh reconstructions by four methods and ground truth.

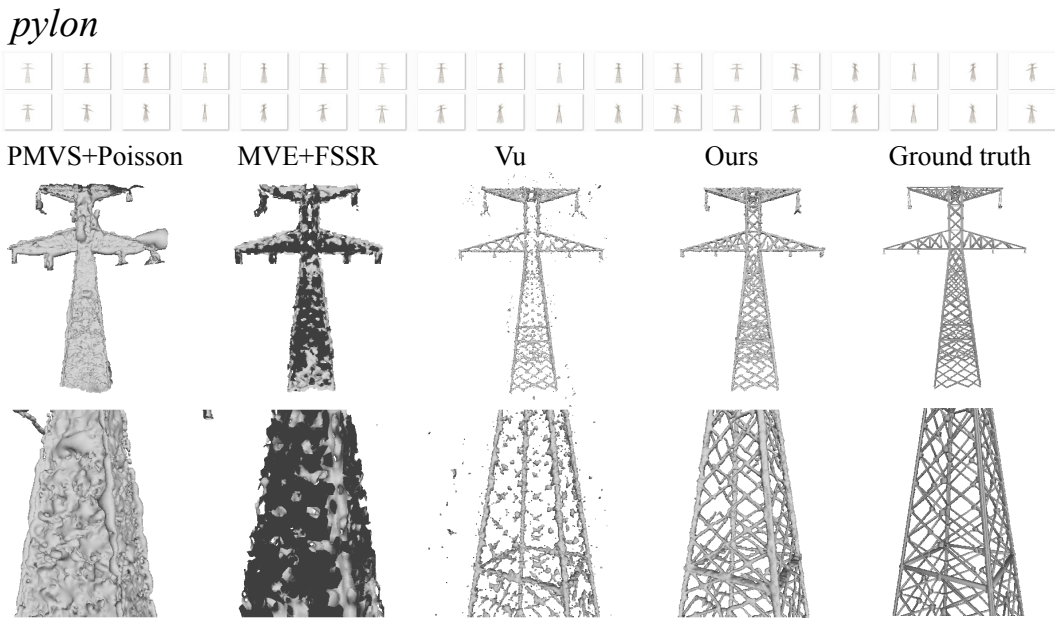


Figure 6: The *pylon* dataset. Upper: input images. Lower: mesh reconstructions by four methods and ground truth.

## 2.2. Real-world datasets

We have shown partial qualitative results in paper Section 4.2. Here we show all six real-world datasets. The names of dataset, number of images and image resolution are shown in Table 1. Here we only present the results of *Vu* [12] method (baseline) and our method to emphasis our improvement. The other methods (*PMVS+Poisson* and *MVE+FSSR*) perform poorly on these datasets and produce severely flipped triangle normals. The intermediate point cloud and reconstructed curves are shown. Note that both methods use the same set of point cloud, so the qualitative difference illustrates the effectiveness of curves.

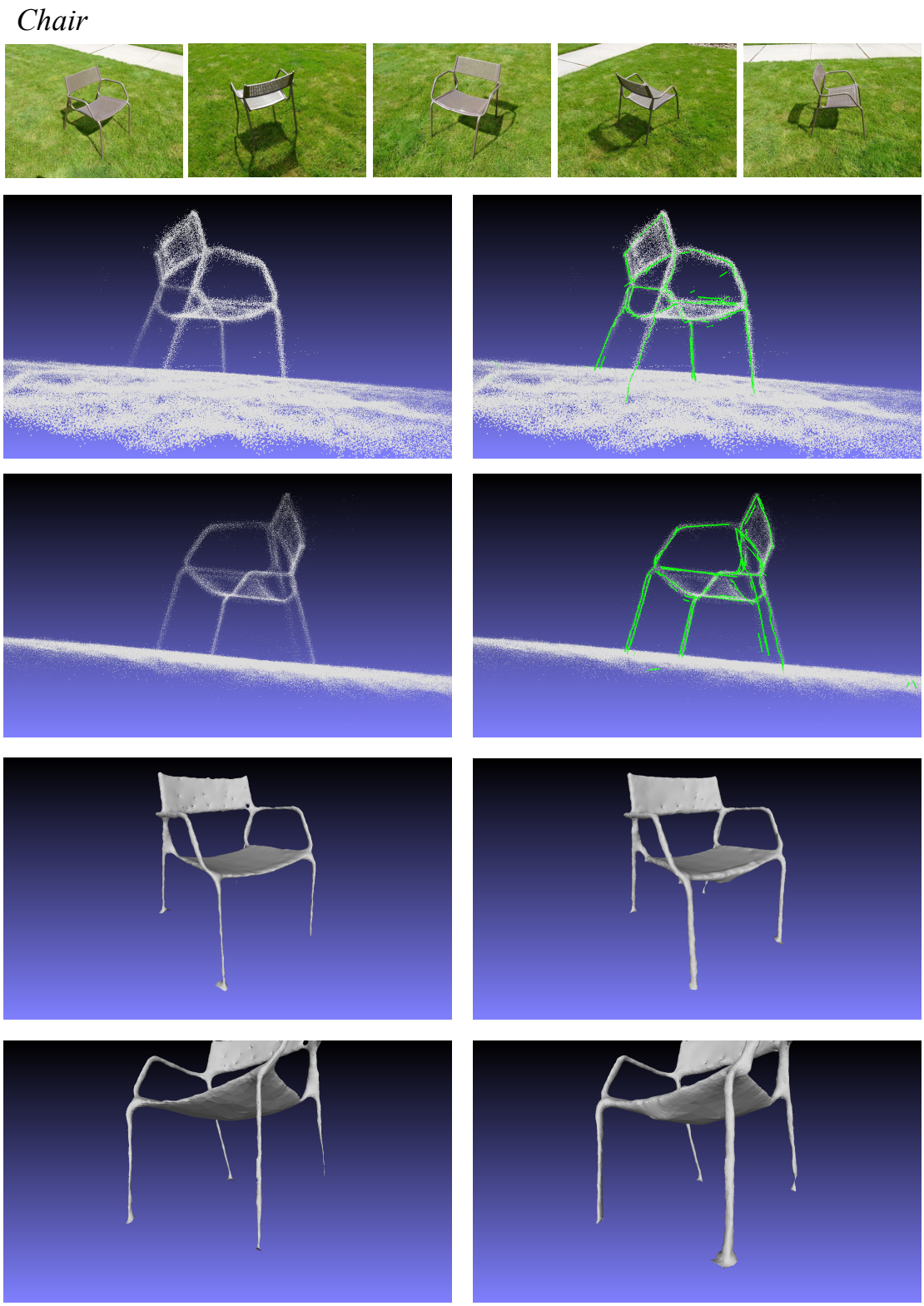


Figure 7: Top: samples of input images. Left column: point cloud and mesh generated by Vu's method [12]. Right column: point cloud with our reconstructed curves (green) and mesh reconstruction by our method.

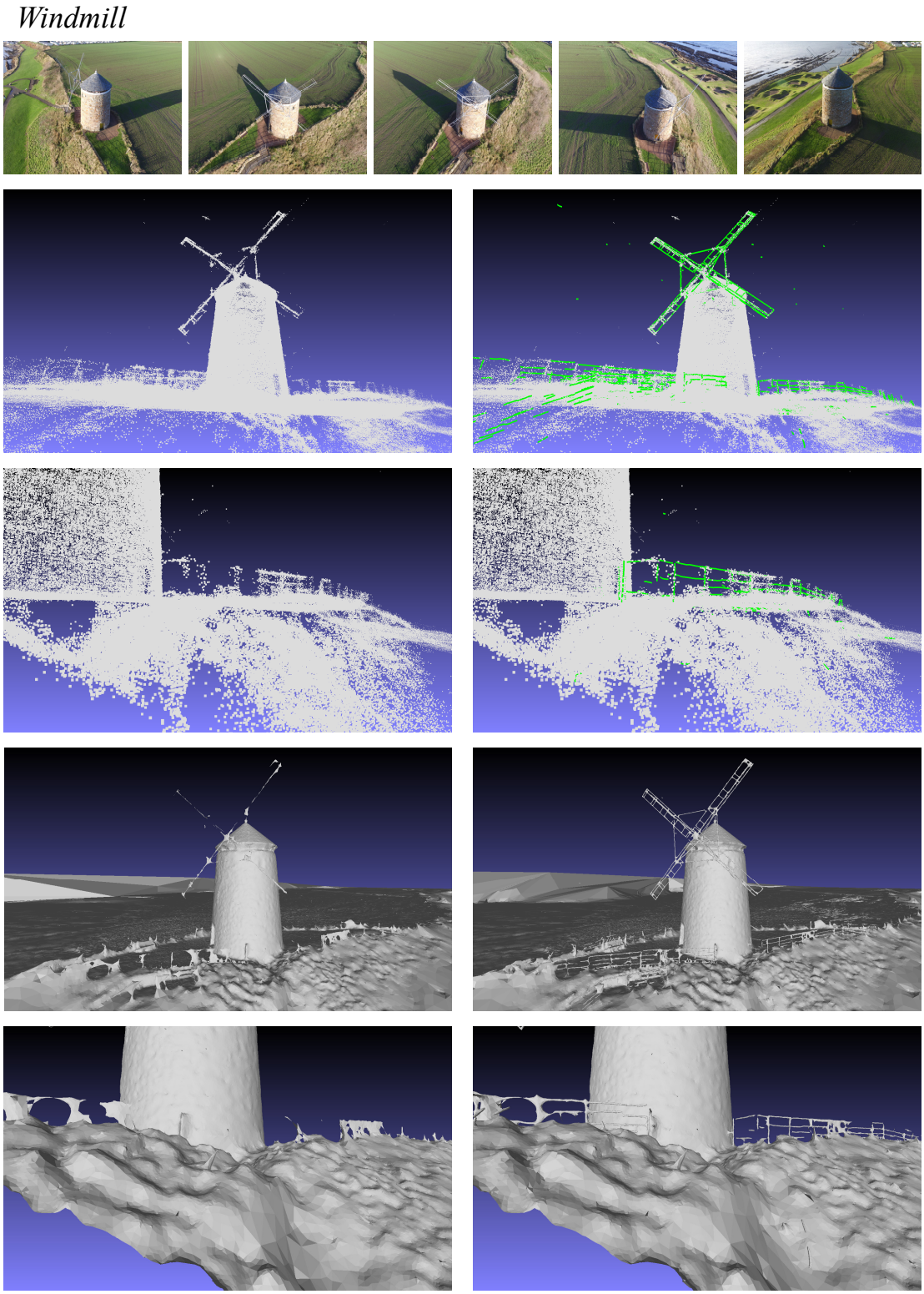


Figure 8: Top: samples of input images. Left column: point cloud and mesh generated by Vu's method [12]. Right column: point cloud with our reconstructed curves (green) and mesh reconstruction by our method.

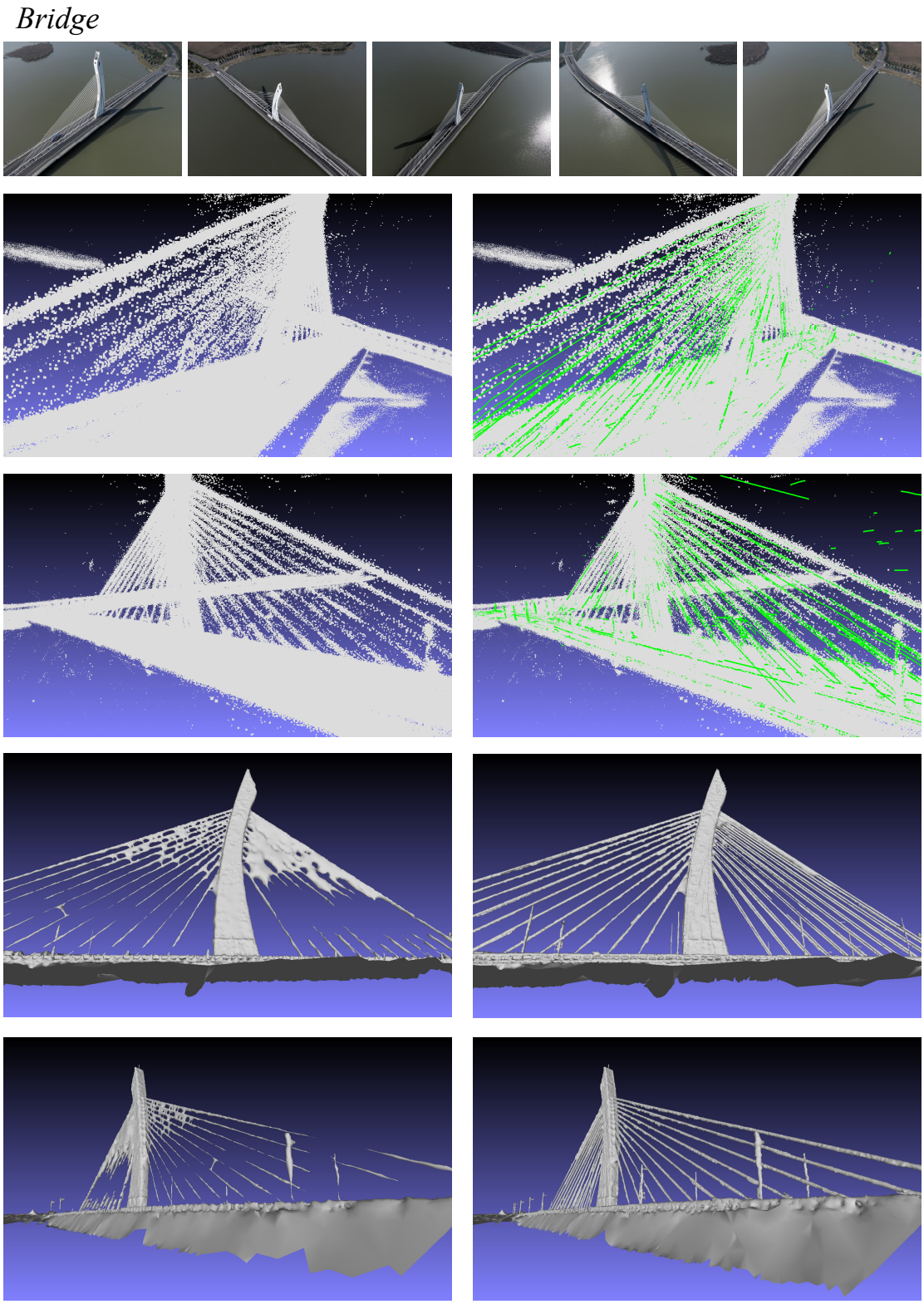


Figure 9: Top: samples of input images. Left column: point cloud and mesh generated by Vu's method [12]. Right column: point cloud with our reconstructed curves (green) and mesh reconstruction by our method.

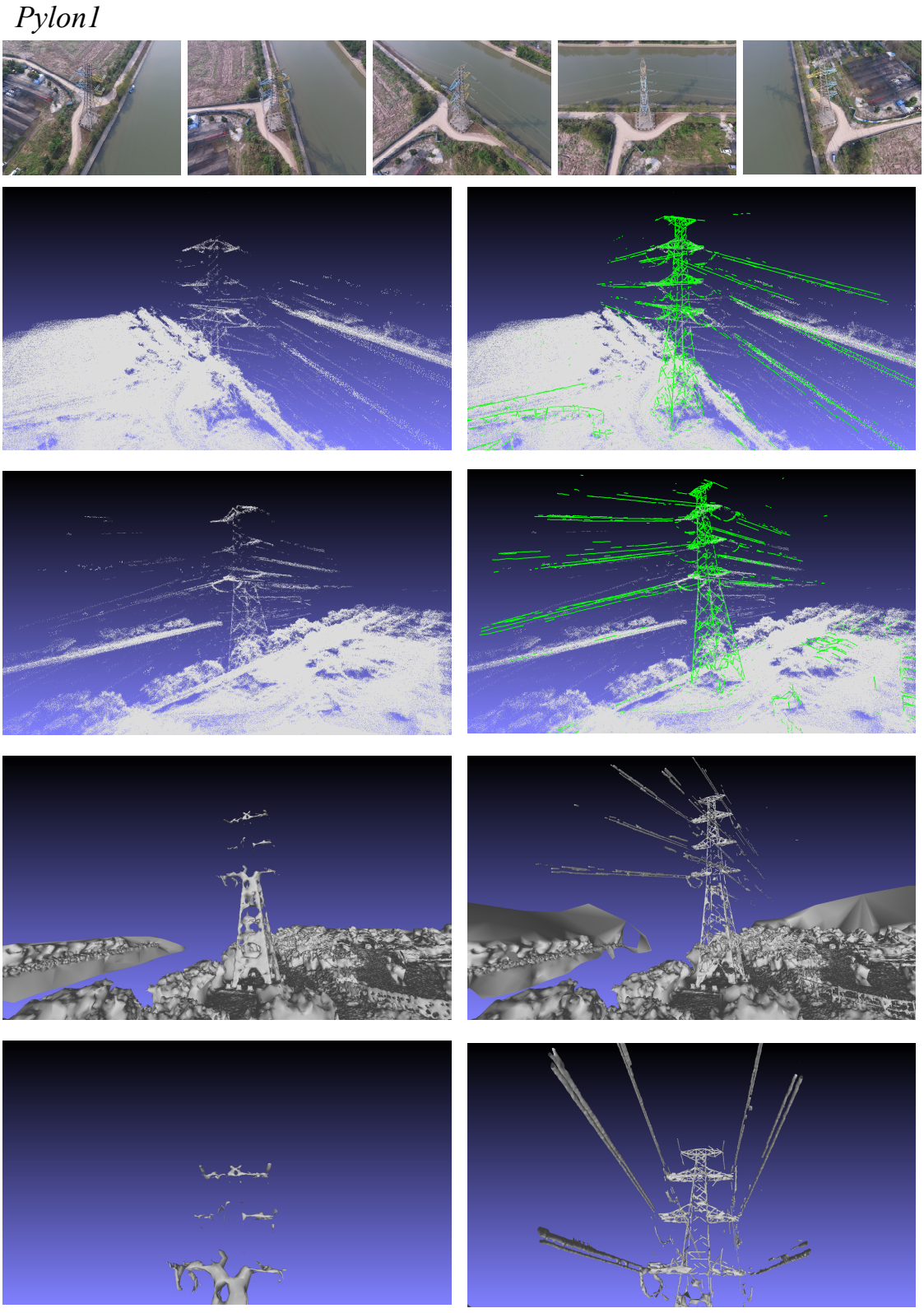


Figure 10: Top: samples of input images. Left column: point cloud and mesh generated by Vu's method [12]. Right column: point cloud with our reconstructed curves (green) and mesh reconstruction by our method.

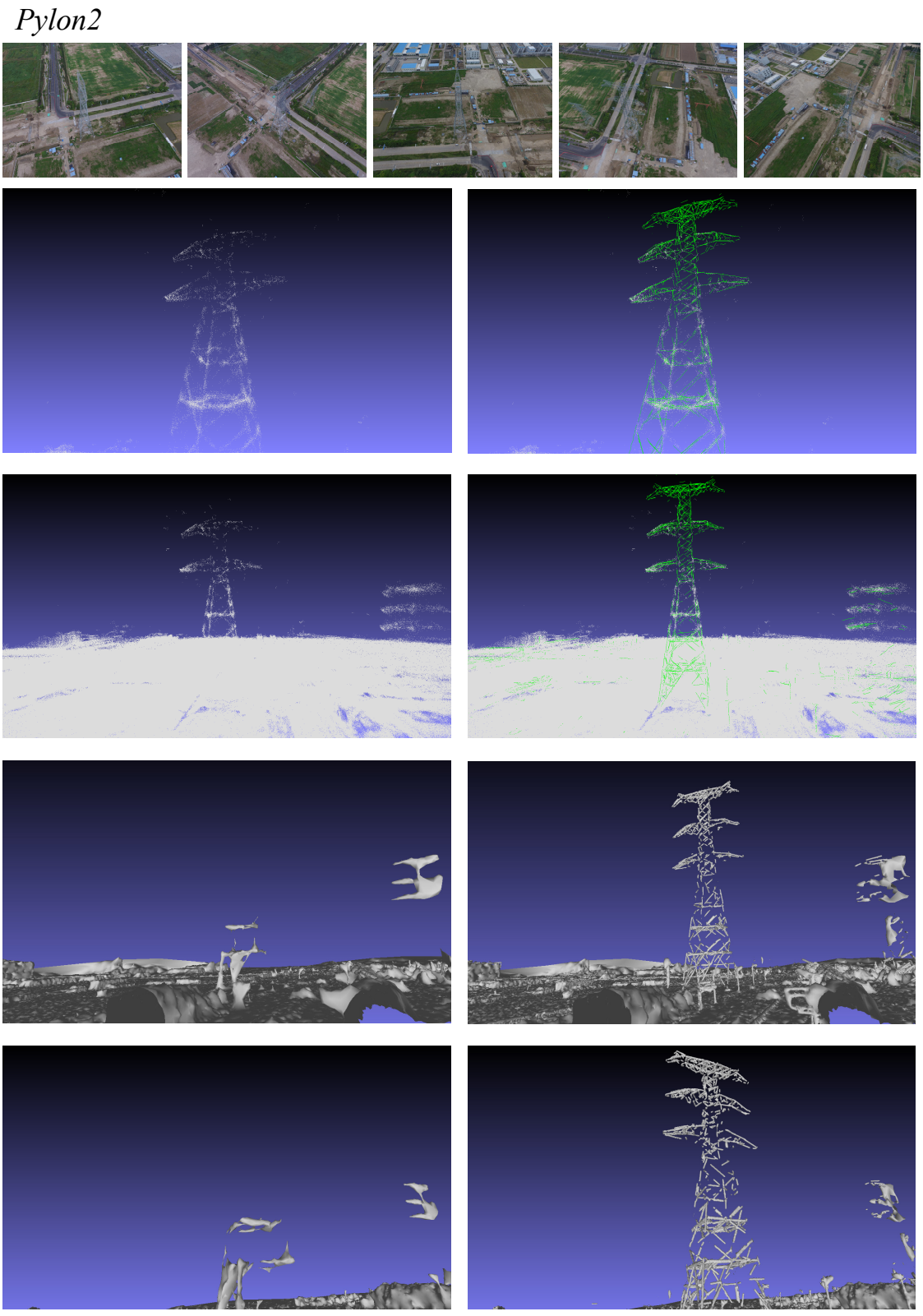


Figure 11: Top: samples of input images. Left column: point cloud and mesh generated by Vu's method [12]. Right column: point cloud with our reconstructed curves (green) and mesh reconstruction by our method.

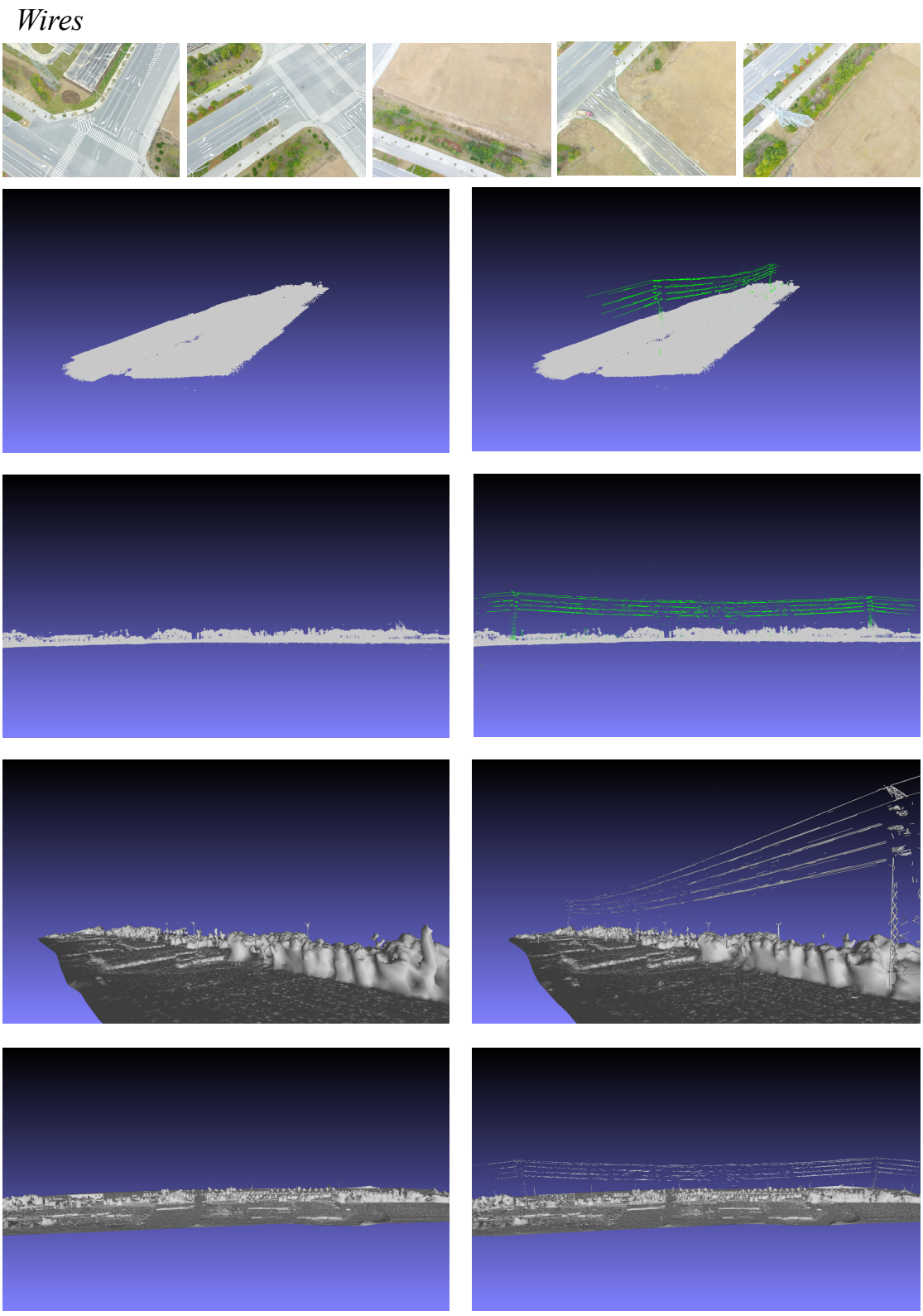


Figure 12: Top: samples of input images. Left column: point cloud and mesh generated by Vu's method [12]. Right column: point cloud with our reconstructed curves (green) and mesh reconstruction by our method.

## 3. EPFL datasets

The proposed method aims at improving thin structure reconstruction, but it can be applied to general datasets without worsening the original results. Here we apply the proposed method to three EPFL datasets [9]. Although these scenes do not contain many thin structures, and most reconstructed curves are textures or shadings on planes, these curves will be fused into the mesh adaptively. The results are shown in Fig. 13, Fig. 14 and Fig. 15.

*Herzjesu_25*

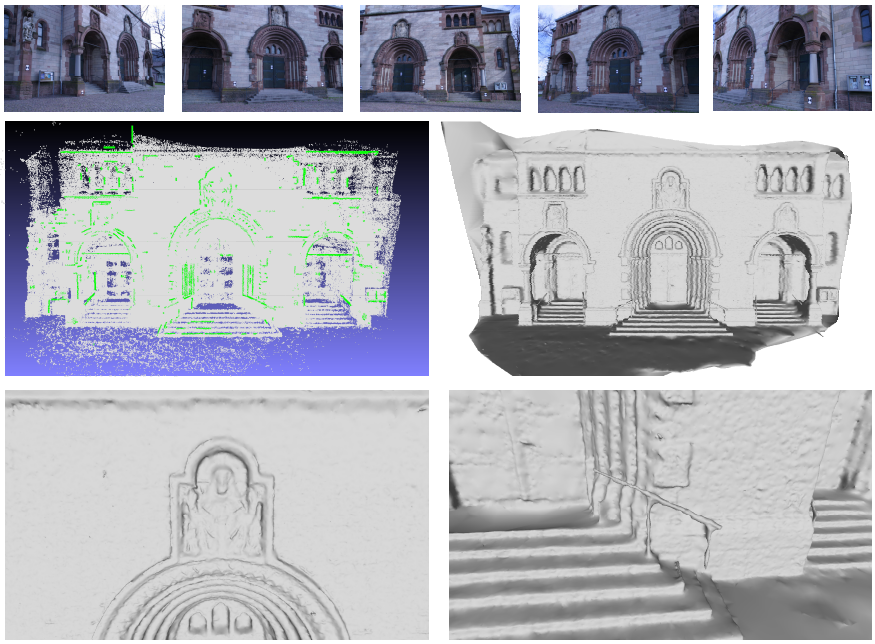

Figure 13: Sample images, point cloud and curves (green), reconstructed mesh.

*Fountain_8*

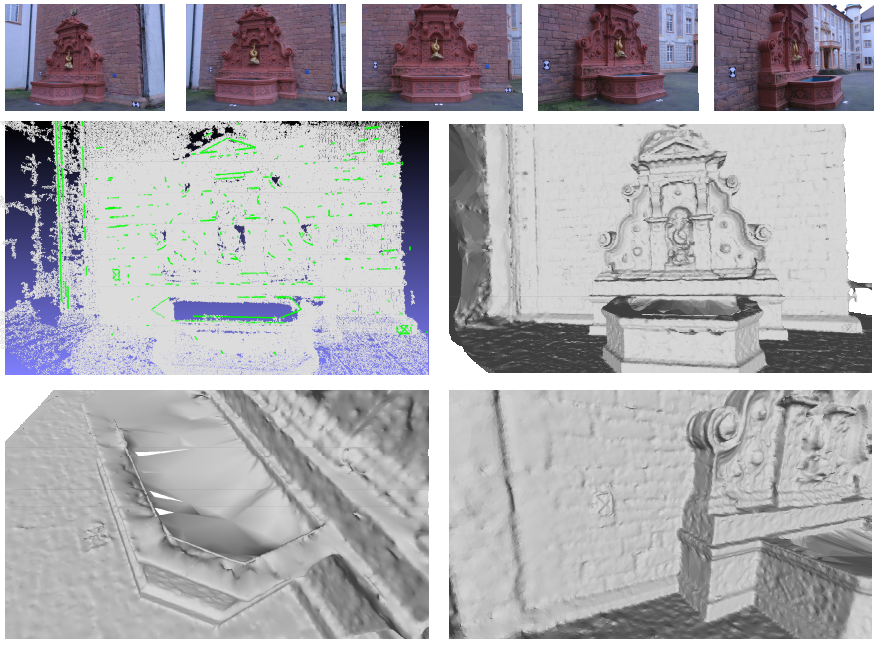

Figure 14: Sample images, point cloud and curves (green), reconstructed mesh.

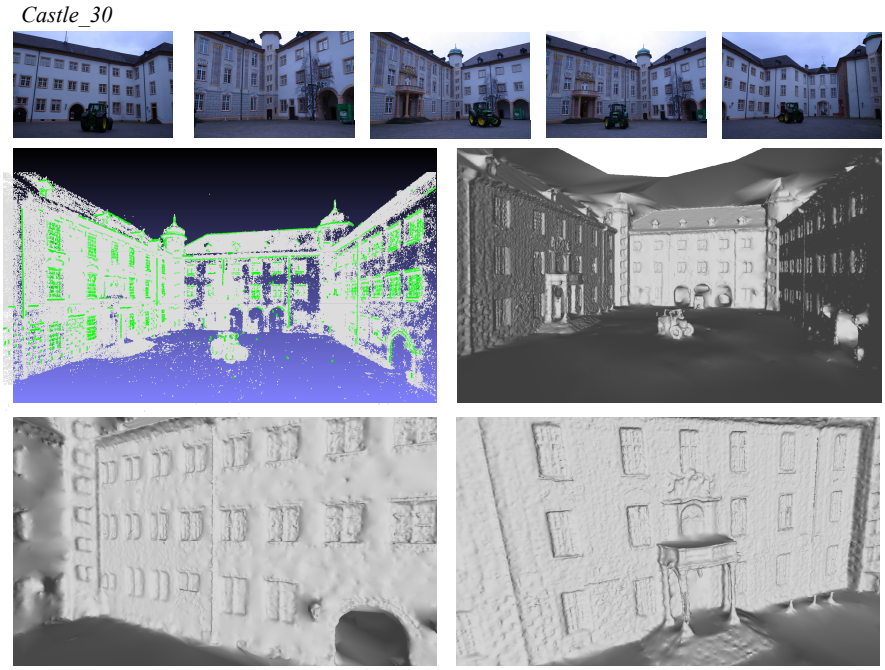


Figure 15: Sample images, point cloud and curves (green), reconstructed mesh.

## 4. Breakdown experiment

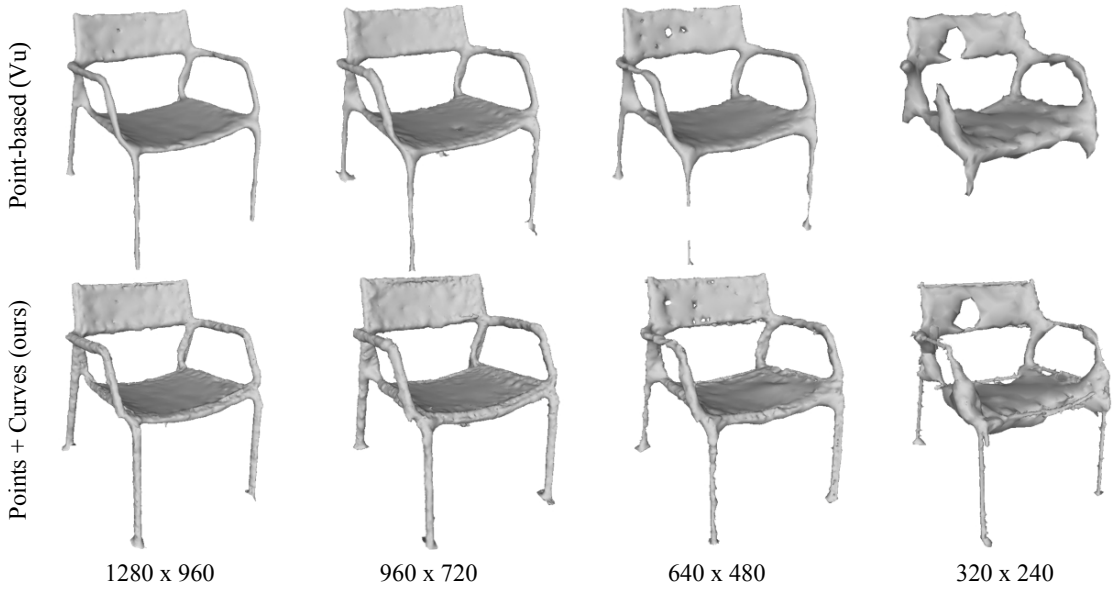


Figure 16: As the image resolution goes down, our method has stronger ability to recover thin structures such as armrests and legs. Here we also improved the point-based method [12] with the *soft-visibility* term [4] to reduce the holes at backrests.

The proposed method is compared with a baseline method [12] by progressively lowering the image resolution, illustrated in Fig. 16. As discussed in the main paper, the thickness of the thin structure is relative to image resolution. This experiment simulates the case when the image resolution is low, or equivalently, the geometry structure is very thin. The result indicates that, the point-based method breaks down at $640 \times 480$ resolution, while our method has stronger ability to recover thin structures such as armrests and legs at low image resolution. Also, the 3D curve at the silhouette of backrest improves the reconstruction of the edges of backrest.

## 5. Evaluation of the negative influences

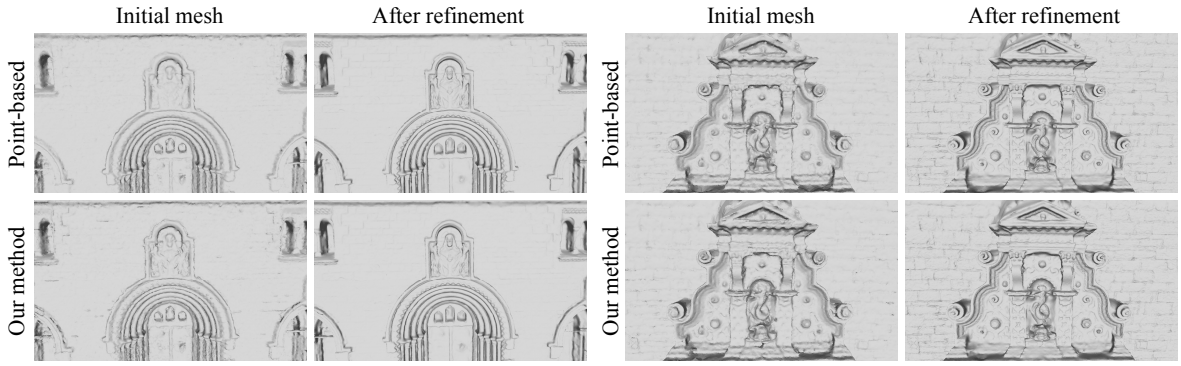


Figure 17: The 3D curves reconstructed on sharp textures might introduce slight bumps due to inaccurate estimated positions of 3D curves or overestimated confidence regions, but these artifacts are very minor and can be easily removed by mesh refinement [12].

As mentioned in the main paper, the reconstructed 3D curves are of three types. For those curves on sharp textures, they imply no geometry and should be fused into the surface. One may suspect these 3D curves may introduces errors to the reconstructed surface as they would look more bumpy than the real geometry is. In our experience, this side effect is minor, possibly due to inaccurate estimated positions of 3D curves or overestimated confidence regions. Most of these artifacts are non-topology flaws, but it may happen when the point cloud is so sparse that the curve of sharp edges induces topological flaws (example in Fig. 14 bottom left). For most inaccuracies such as slight bumps or inflations over the textured edges, a following mesh refinement can correct them.

Here, we conduct an experiment to evaluation the negative influence brought by the 3D curves. We use the data of *fountain* and *herjesu* in [9] as the ground truth are provided. We reconstruct the surface w/ and w/o the proposed 3D curves, and evaluate the difference before and after refinement. The results are visualized in Fig. 17. In the initial meshes, the mesh produced by our method has slight bumps on the sharp texture. After refinement, these artifacts are eliminated and visually the same with the refined mesh of point-based method. We also tried to measure the distance between reconstructed meshes and ground truth meshes (using the same quantitative evaluation as in the main paper), but it turned out the gap between point-based and our method is extremely minor and inconsistent on two datasets. This finding also conveys that the negative influence brought by 3D curves is very minor and ignorable.

## 6. Comparisons of 3D curve reconstruction

The 3D curve reconstruction (Section 2 in the main paper) can be a standalone component, for the purpose of scene abstraction, edge drawing creation, etc. Here, we compare our method with two recently proposed methods namely, *Line3D* [3], *3D Drawing* [11]. We use the *Vase* dataset (a successful example in [11]'s paper), and create the results of *Line3D* (code from https://github.com/manhofer/Line3D) and ours.

Fig. 18 shows the results of three methods. The 3D drawing [11] generates a complete set of 3D curves. However, they are redundant (one continuous curve is reconstructed by many fragmentary curves), unstructured and visually noisy. *Line3D* [3] approximates the 3D curves with line segments and fails to represent accurate geometries. Our result enjoys a decent balance between accuracy and completeness by visual evaluation. The comparisons with other related works (*e.g.*, *Edge-based SfM* [5], *Sampling-based* [10]) are currently not available due to the lack of open-source implementation.

## 7. Further explanations of Curve-conformed Delaunay Refinement

The Curve-conformed Delaunay Refinement presented in the main paper Section 3.2 involves computational geometry background knowledges. As they are out of the scope of the paper, we only put minimal and self-contained content in the paper. Here, we further explain the backgrounds of proposed algorithm. For more details, we refer readers to [6, 7, 8].

The proposed Curve-conformed Delaunay Refinement is inspired by two previous Delaunay Refinement algorithms, namely *Ruppert's algorithm* [6] and *Shewchuk's algorithm* [7]. Here we briefly introduce their algorithms.

*Ruppert's algorithm* [6] is proposed initially for quality 2D Delaunay mesh generation. The input of their algorithm is *planar straight line graph* (PSLG) and the output is a quality Delaunay mesh conforming to the PSLG. There are four

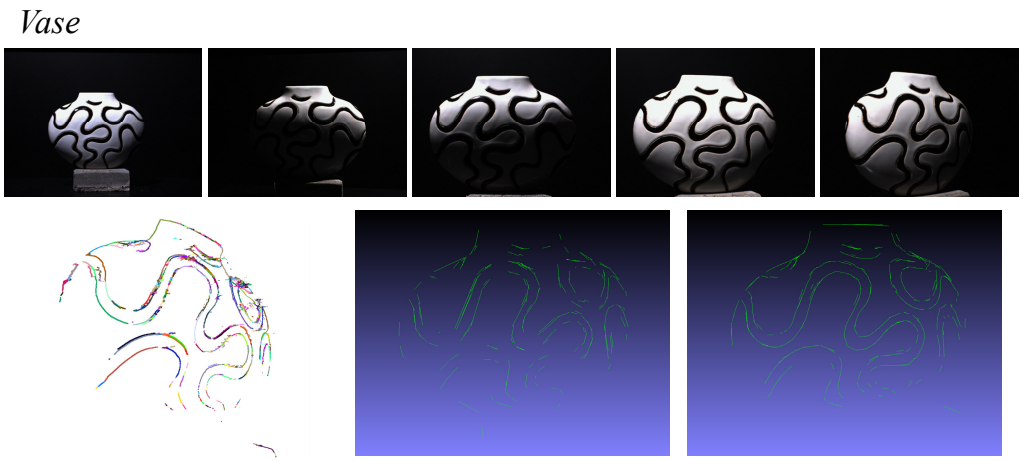


Figure 18: Results of *Vase* dataset [**?**]. Left: 3D drawing result [11]. Mid: *Line3D* [3] result. Right: our result.

stages in the algorithm. 1) They first apply Delaunay triangulation on vertices of PSLG, momentarily ignoring the segment connections. 2) The second stage is to do the *segment recovery*: for any segment in PSLG that does not existed in the Delaunay triangulation, a midpoint vertex of that segment is inserted to the Delaunay triangulation, until all segments of PSLG exist in the Delaunay mesh. 3) The third stage is simply removing triangles from concavities and holes (*i.e.*, outside the interior region of PSLG). 4) The fourth stage is the key the algorithm: the Delaunay mesh is iteratively refined by inserting vertices. The vertex insertion is governed by two rules: a vertex in the midpoint of segment is added if the segment's *diametral circle* encloses any other vertex, and, a vertex in the circumcircle of a triangle is added if the triangle is badly-shaped.

*Shewchuk's algorithm* [7] generalized the *Ruppert's algorithm* into three-dimensional tetrahedra. The input becomes *piecewise linear complex* (PLC) (instead of 2D PSLG) and output is a quality tetrahedra conforming to the input PLC. This algorithm has three stages only. Their first and third stages are similar to *Ruppert's algorithm*, but they combine all vertex insertion operations into the second stage. Concretely, 1) they first apply Delaunay tetrahedralization on vertices of PLC, momentarily ignoring segments and facets. 2) The second stage is the key of algorithm: adding the vertex to refine the tetrahedra by three rules. R1: adding vertex at the midpoint of a segment if its diametric sphere encloses any other vertex. R2: adding vertex at the center of a diametric sphere of a triangle facet, if the diametric sphere encloses any other vertex. R3: adding vertex at the center of a tetrahedron if it is badly-shaped. The priority orders from R1, R2 to R3. The specific tetrahedron shape measurement is defined as *radius-edge ratio*, which is the ratio between the radius of its circumsphere and its shortest edge. 3) The third stage is simply removing tetrahedra from concavities and holes (*i.e.*, outside the interior region of PLC). This algorithm is provably good with a *radius-edge ratio* threshold value of 2 [7].

The above two algorithms are general algorithms for quality Delaunay triangulation/tetrahedra generation. Our proposed algorithm can be seen as a variant of above two algorithms. Our three criteria are similar to three rules of vertex insertion used in *Shewchuk's algorithm*. However, we do not need to apply the vertex insertion for facets (R2) as our input is a bunch of segments only. Other than that, we have another criterion that constrains the volume size by clamping the edge length of segments, which is necessary for thin structure recovery as our goal.

## 8. Runtime performance

The runtime performance of six qualitative datasets is shown in Table 1. We separately evaluate two main proposed components. Our *3D curve reconstruction* (Section 2 in the main paper) is compared to two point cloud methods, namely *plane sweep stereo* [2] and *PMVS* [1] and the *Line3D* method [3]. *Line3D* method [3] is the initialization of our method and we view it as our baseline. The *manifold surface reconstruction* (Section 3 in the main paper) is compared to the baseline *visibility consistent surface reconstruction* [12].

The main computation of our *3D curve reconstruction* spends on the *Line3D* initialization. As can be seen in Table 1, our algorithm spends $7\sim25\%$ more running time than baseline method, which accounts for the optimization and expansion steps. As for our *manifold surface reconstruction* step, it has the same theoretical complexity as the baseline method, with additional vertices (generated in the Delaunay refinement) in tetrahedra. The actual increased time is $10\sim28\%$. Overall, the

Table 1: Running time (unit: seconds) on six datasets. The "increased time (%)" means running time of our method compared to baseline.

| Dataset | #img | resolution | PMVS [1] | Sweep [2] | Lines [3] (baseline) | Curves (ours) | Increased time (%) | Vu [12] (baseline) | Ours | Increased time (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| *chair* | 41 | 1280 x 960 | 62 | 18 | 6 | 7 | 16.67% | 25 | 32 | 28.00% |
| *windmill* | 27 | 4000 x 3000 | 457 | 217 | 84 | 103 | 22.62% | 120 | 142 | 18.33% |
| *pylon1* | 32 | 4000 x 3000 | 512 | 230 | 96 | 120 | 25.00% | 123 | 145 | 17.89% |
| *pylon2* | 54 | 4000 x 3000 | 991 | 408 | 120 | 141 | 17.50% | 182 | 207 | 13.74% |
| *bridge* | 117 | 4000 x 3000 | 2724 | 1440 | 280 | 332 | 18.57% | 318 | 358 | 12.58% |
| *wires* | 296 | 4000 x 3000 | 7538 | 2988 | 574 | 606 | 7.45% | 524 | 579 | 10.50% |

runtime performance of proposed method is reasonable.

## References


[1] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[2] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[3] M. Hofer, M. Maurer, and H. Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding*, 157:167–178, 2017.

[4] P. Labatut, J.-P. Pons, and R. Keriven. Robust and efficient surface reconstruction from range data. In *Computer graphics forum*, volume 28, pages 2275–2290. Wiley Online Library, 2009.

[5] I. Nurutdinova and A. Fitzgibbon. Towards pointless structure from motion: 3d reconstruction and camera parameters from general 3d curves. In *International Conference on Computer Vision (ICCV)*, pages 2363–2371. IEEE, 2015.

[6] J. Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of algorithms*, 18(3):548–585, 1995.

[7] J. R. Shewchuk. Delaunay refinement mesh generation. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1997.

[8] H. Si. Three dimensional boundary conforming delaunay mesh generation. 2008.

[9] C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[10] D. Teney and J. Piater. Sampling-based multiview reconstruction without correspondences for 3d edges. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 160–167. IEEE, 2012.

[11] A. Usumezbas, R. Fabbri, and B. B. Kimia. From multiview image curves to 3d drawings. In *European Conference on Computer Vision (ECCV)*, pages 70–87. Springer, 2016.

[12] H.-H. Vu, P. Labatut, J.-P. Pons, and R. Keriven. High accuracy and visibility-consistent dense multiview stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):889–901, 2012.