Supplementary material for "Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds"

Yujia Liu
ETH Zürich

Anton Obukhov
ETH Zürich

Jan Dirk Wegner
University of Zürich

Konrad Schindler
ETH Zürich

A1. Notation

Throughout the paper, we use several terms and concepts as follows. A **CAD model** represents a complex object in a format that permits editing in CAD software. **Boundary representation (B-REP)** is one such representation; it involves analytic **surfaces**, **edges**, and **corners**[1], and explicit topological relationships between them. Typically, such relationships take the form of "an edge is an intersection of two surfaces" and "a corner is an intersection of two edges", stated as adjacency matrices. The manifolds of surfaces and edges have intrinsic dimensionality of 2 and 1 respectively. **Point cloud** refers to the unordered set of 3D points, obtained by scanning the surface of a real world object or by sampling a CAD model for simulation purposes. **Semantic classes** are the geometric types that we consider for representing CAD model surfaces. These typically include a plane, sphere, cylinder, cone, and freeform (spline) surfaces. **Segmentation** refers to assigning a semantic class to each point of the point cloud. **Clusters** are sets of points belonging to the same semantic class and instance; they group points belonging to distinct surfaces. **Clustering** information can be a byproduct of segmentation, an algorithmic stage, or given as ground truth. **Primitives** are inherently low-dimensional parametric models representing analytical shapes of semantic classes, and that can be used to fit clusters with least squares.

A2. Primitive parameterization and fitting

Our method can handle several primitive types, including planes, spheres, cylinders, and cones [1, 2].

Plane In the context of 3D geometry, a plane can be represented by a vector $\mathbf{n} \in \mathbb{R}^3$, which is the unit normal to the plane, and a scalar d that determines the distance of the plane from the origin. Mathematically, a plane can be denoted by a tuple $(\mathbf{n}, d)$, where $\mathbf{n}$ satisfies $|\mathbf{n}| = 1$. Any point $\mathbf{p} \in \mathbb{R}^3$ lies on the plane if and only if $\mathbf{n}^{\mathrm{T}}\mathbf{p} = d$.

The following steps can be employed to perform plane fitting to a collection of 3D points:

1. Calculate the centroid of the points: $\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p_i}$;
2. Compute the matrix: $\mathbf{M} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{p_i} - \mathbf{c})(\mathbf{p_i} - \mathbf{c})^T$;
3. Calculate the eigenvalues and eigenvectors of $\mathbf{M}$. Select the eigenvector corresponding to the smallest eigenvalue as the normal vector $\mathbf{n}$. Compute the distance d using the normal vector $\mathbf{n}$ and the centroid c: $d = \mathbf{n}^T\mathbf{c}$.

Sphere A sphere can be represented by a tuple $(\mathbf{c}, r)$, where $\mathbf{c} \in \mathbb{R}^3$ denotes the center of the sphere and $r \in \mathbb{R}$ denotes its radius. Any point $\mathbf{p} \in \mathbb{R}^3$ lies on the sphere if and only if $||p-\mathbf{c}|| = r$. The error function to be minimized is as follows:

$$E\left(\mathbf{c}, r\right) = \sum_{i=1}^{M} \left(\|\mathbf{p}_i - \mathbf{c}\| - r\right)^2. \tag{1}$$

Solving $\frac{\partial E}{\partial r} = 0$ for r yields

$$r = \frac{1}{M}\sum_{i=1}^{M} \|\mathbf{p}_i - \mathbf{c}\|. \tag{2}$$

Solving $\frac{\partial E}{\partial \mathbf{c}} = 0$ for $\mathbf{c}$ gives

$$\mathbf{c} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{p}_i + r \cdot \frac{1}{M}\sum_{i=1}^{M} \frac{\partial(\|\mathbf{p}_i - \mathbf{c}\|)}{\partial \mathbf{c}}. \tag{3}$$

A fixed point iteration can solve the equations and subsequently determine the parameters $(\mathbf{c}, r)$.

Cylinder An infinite cylinder $(\mathbf{a}, \mathbf{c}, r)$ is characterized by a point $\mathbf{c}$, a unit-length direction vector $\mathbf{a}$ that defines its axis and a radius r. Any point $\mathbf{p} \in \mathbb{R}^3$ lies on the cylinder if and only if $(\mathbf{p} - \mathbf{c})^T \left(\mathbf{I} - \mathbf{a}\mathbf{a}^T\right)(\mathbf{p} - \mathbf{c}) = r$ The error function to be minimized is

$$E\left(\mathbf{a}, \mathbf{c}, r\right) = \sum_{i=1}^{M} \Big((\mathbf{p}_i - \mathbf{c})^T \left(\mathbf{I} - \mathbf{a}\mathbf{a}^T\right)(\mathbf{p}_i - \mathbf{c}) - r^2 \Big)^2, \tag{4}$$

[1] Related terms in literature: "surfaces" := "faces", "patches"; "edges" := "curves", "contours"; "corners" := "points", "vertices", "endpoints".

where $\mathbf{I}$ is the identity matrix. Solving $\frac{\partial E}{\partial r^2} = 0$ leads to

$$r = \left(\frac{1}{M} \sum_{i=1}^{M} (\mathbf{c} - \mathbf{p}_i)^T (\mathbf{I} - \mathbf{a}\mathbf{a}^T)(\mathbf{c} - \mathbf{p}_i) \right)^{\frac{1}{2}}. \quad (5)$$

By solving $\frac{\partial E}{\partial \mathbf{c}} = 0$, we can obtain an equation for $\mathbf{c}$:

$$\mathbf{c} = \frac{\tilde{\mathbf{A}}^{-1}\mathbf{A}}{\mathrm{tr}(\hat{\mathbf{A}}\mathbf{A})} \left(\frac{1}{M} \sum_{i=1}^{M} \left(\mathbf{p}_i^T \tilde{\mathbf{A}} \mathbf{p}_i \right) \mathbf{p}_i \right), \quad (6)$$

where

$$\mathbf{A} = \tilde{\mathbf{A}} \left(\frac{1}{M} \sum_{i=1}^{M} \mathbf{p}_i \mathbf{p}_i^T \right) \tilde{\mathbf{A}}, \quad (7)$$

and $\hat{\mathbf{A}} = \mathbf{S}\mathbf{A}\mathbf{S}^T$, $\tilde{\mathbf{A}} = \mathbf{I} - \mathbf{a}\mathbf{a}^T$, $\mathbf{S}$ is the skew symmetric matrix of $\mathbf{a}$. Putting them back into the error function yields:

$$G(\mathbf{a}) = \frac{1}{M} \sum_{i=1}^{M} \Bigg[\mathbf{p}_i^T \tilde{\mathbf{A}} \mathbf{p}_i - \frac{1}{M} \sum_{j=1}^{M} \mathbf{p}_j^T \tilde{\mathbf{A}} \mathbf{p}_j - \\ - \frac{2\mathbf{p}_i^T \hat{\mathbf{A}}}{\mathrm{Tr}\left(\hat{\mathbf{A}}\mathbf{A}\right)} \left(\frac{1}{M} \sum_{j=1}^{M} \left(\mathbf{p}_j^T \tilde{\mathbf{A}} \mathbf{p}_j \right) \mathbf{p}_j \right) \Bigg]^2. \quad (8)$$

A Powell optimizer [3] is then employed to locate the global minimum of the function $G(\mathbf{a})$. Once $\mathbf{a}$ has been determined, the center $\mathbf{c}$ and radius r of the corresponding circle can be easily obtained through Eq. 5 and 6.

Cone We parameterize an infinite cone using the set of parameters $(\mathbf{v}, \mathbf{a}, \theta)$, where $\mathbf{v} \in \mathbb{R}^3$ denotes the apex point, $\mathbf{a} \in \mathbb{R}^3$ denotes a unit axis direction vector, and $\theta \in (0, \pi/2)$ represents half the angle of the cone. Any point $\mathbf{p}$ that lies on it satisfies $\mathbf{a} \cdot \frac{\mathbf{p}-\mathbf{v}}{||\mathbf{p}-\mathbf{v}||} = \cos(\theta)$, which can be written in a quadratic form as $(\mathbf{p} - \mathbf{v})^T (\cos(\theta)^2 \mathbf{I} - \mathbf{a}\mathbf{a}^T)(\mathbf{p} - \mathbf{v}) = 0$. Thus, the error function can be defined as follows:

$$E(\mathbf{v},\mathbf{a},\theta) = \sum_{i=1}^{M} \left((\mathbf{p}_i - \mathbf{v})^T \left(\cos(\theta)^2 \mathbf{I} - \mathbf{a}\mathbf{a}^T \right) (\mathbf{p}_i - \mathbf{v}) \right)^2. \quad (9)$$

This least-square problem can be solved efficiently with the Levenberg-Marquardt [4] method.

A3. Algorithm of topological reconstruction

We present a pseudocode-based methodology for reconstructing topological structures derived from M surfaces fitted with potentially infinite primitives, forming a set $\{\mathbf{S}_k^0\}_M$. We denote the stage of surface processing with superscript. We trim each primitive to form a margin of width ϵ around the input points, then employ tessellation and triangulation meshing algorithm. As a result, we obtain a set of finite extended surfaces denoted as $\{\mathbf{S}_k^1\}_M$. We generate poly-line edges $\{\mathbf{E}_k^1\}_K$ by identifying intersecting surfaces and computing pairwise intersections. And we trim the surfaces by the edges, thus obtaining $\{\mathbf{S}_k^2\}_M$. Similarly, we intersect adjacent poly-line edges to obtain corner points $\{\mathbf{C}_k\}_L$, subsequently trimming edges accordingly to get final edges $\{\mathbf{E}_k^2\}_K$. See Alg. 1.

Algorithm 1 Point2CAD model reconstruction

Input: M surfaces $\{\mathbf{S}_k^0\}_M$ obtained from fitting primitives to point clusters $\{\mathbf{P}_k\}_M$
Output: a CAD model with M surfaces $\{\mathbf{S}_k^2\}_M$, K edges $\{\mathbf{E}_k^2\}_K$ and L corners $\{\mathbf{C}_k\}_L$

for $i \in 1..M$ **do**
  Trim $\mathbf{S}_i^0$ by ϵ to input points: $\mathbf{S}_i^1 = \mathrm{Trim}(\mathbf{S}_i^0 | \mathbf{P}_i, \epsilon)$
end for
for $i \in 1..M$ **do**
  get edges on $\mathbf{S}_i^1$: $\{\mathbf{E}_{i,r}^1\}_{R_i} = \{\mathbf{S}_j^1\}_{j \neq i} \cap \mathbf{S}_i^1$
  trim $\mathbf{S}_i^1$ by edges: $\mathbf{S}_i^2 = \mathrm{Trim}(\mathbf{S}_i^1 | \{\mathbf{E}_{i,r}^1\}_{R_i})$
end for
for each pair $(\mathbf{E}_p, \mathbf{E}_q)$ in intersection edges **do**
  obtain the corners $\mathbf{C}_{pq} = \mathbf{E}_p \cap \mathbf{E}_q$
end for
for $i \in 1..L$ **do**
  if any $\mathbf{C}_s \in \mathbf{E}_i$ **then**
    trim it by the corners: $\mathbf{E}_i^2 = \mathrm{Trim}(\mathbf{E}_i^1 | \mathbf{C}_s)$
  end if
end for

Two distinct cases exist of using the Trim operation on surfaces. In the first case, trimming is performed based on a distance threshold, retaining only the portion of the infinite primitives near the input points. In the second case, we trim the surfaces by considering the intersection of their triangle-mesh representations with edges. We employ connected component analysis, whereby a pair of faces is considered connected if a path exists between them that does not cross an edge obtained through the surface intersection. We then discard whole connected components based on the distance of their members to the original point cloud. The Trim operation on edges involves a similar subdivision of the edge into connected segments by corners and retaining segments close to the input points.

A4. Results on incomplete data

While Point2CAD can operate on top of a generative pipeline, where missing parts of point clouds are reconstructed with the help of a learned prior, ParSeNet and other considered networks have limited ability to work with missing parts of the input. We show examples for incomplete

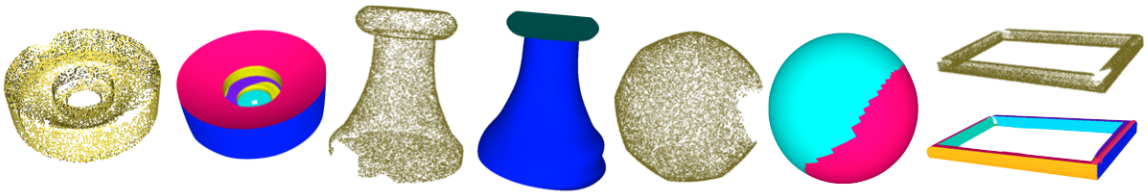

Figure 1. Reconstruction of incomplete point clouds.

point clouds processed by Point2CAD with ParSeNet in Fig. 1.

A5. Results on Real-world 3D scanned data

We show an example for the real scan of a section of railway track reconstructed with our method in Fig. 2. The piece was scanned using a GOM ATOS Core 300, a structured light scanner designed for actual industrial applications.

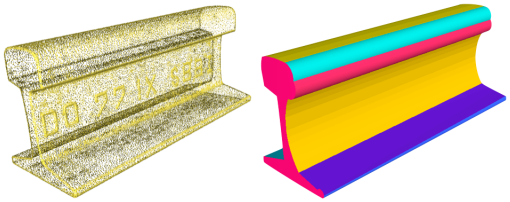

Figure 2. Reconstruction of a real object.

A6. Additional qualitative examples

We show more qualitative examples in Fig. 3, 4, 5, from left to right: (a) input point cloud, (b) ground truth mesh, (c) reconstruction with ComplexGen, (d) Point2CAD with HPNet, (e) Point2CAD with ParSeNet, (f) Point2CAD with GT segmentation.

References

[1] David Eberly. Geometric tools, 2006. 1

[2] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *CVPR*, 2019. 1

[3] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964. 2

[4] Ananth Ranganathan. The levenberg-marquardt algorithm. *Tutoral on LM algorithm*, 11(1):101–110, 2004. 2

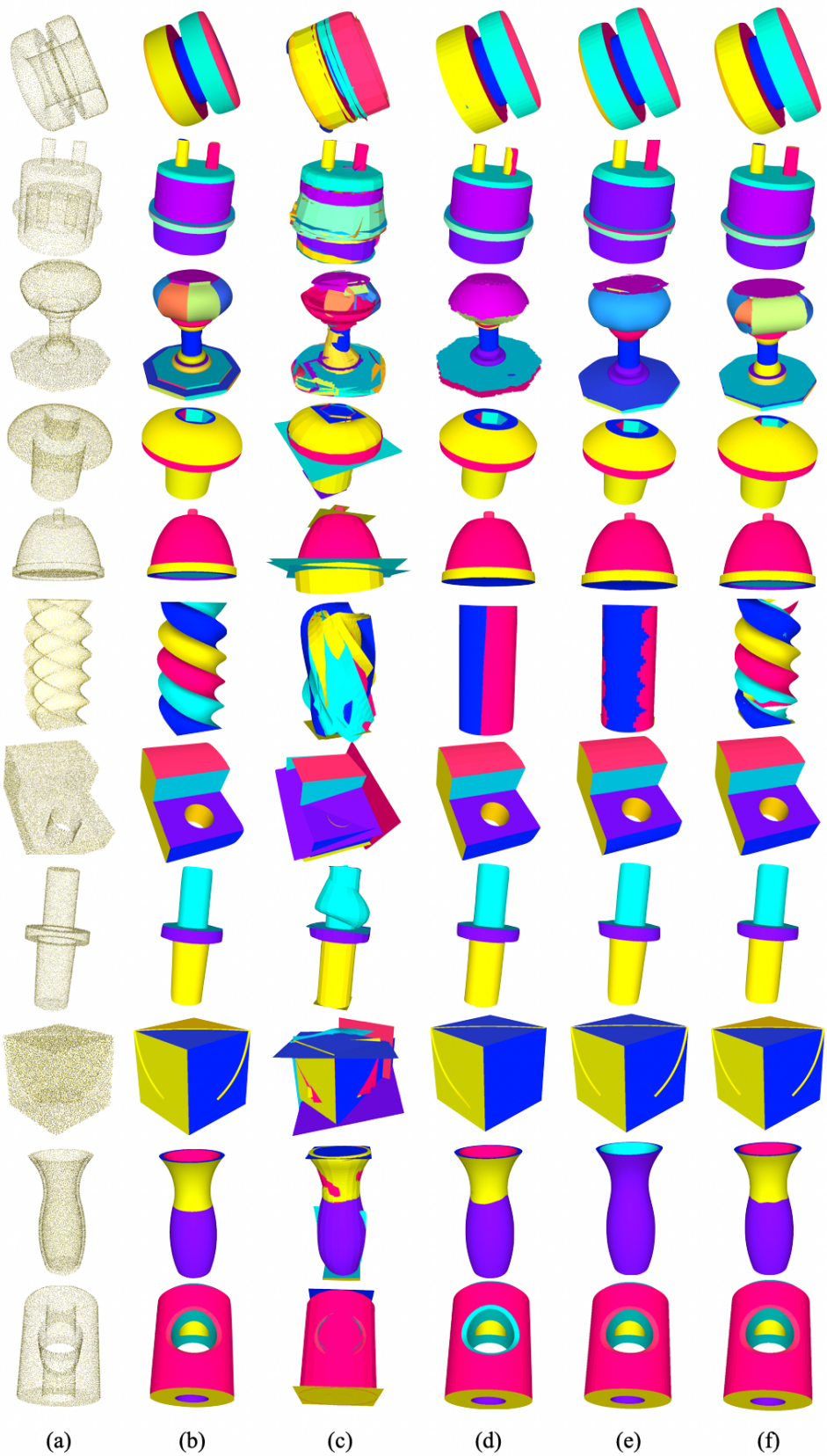

Figure 3. More visualisation results. From left to right: (a) input point cloud, (b) ground truth mesh, (c) reconstruction with ComplexGen, (d) Point2CAD with HP-Net, (e) Point2CAD with ParSeNet, (f) Point2CAD with GT segmentation.

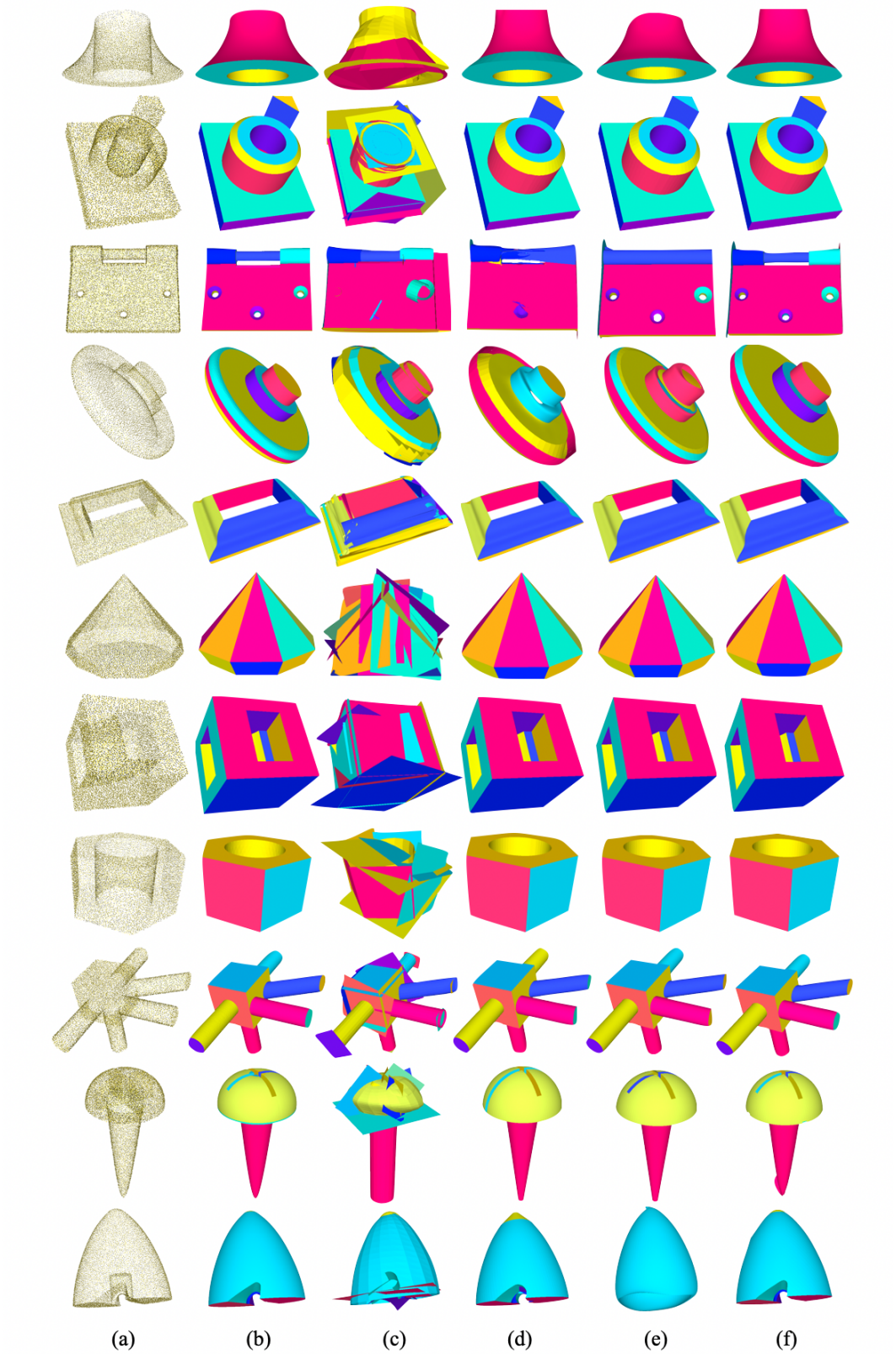

Figure 4. More visualisation results. From left to right: (a) input point cloud, (b) ground truth mesh, (c) reconstruction with ComplexGen, (d) Point2CAD with HP-Net, (e) Point2CAD with ParSeNet, (f) Point2CAD with GT segmentation.

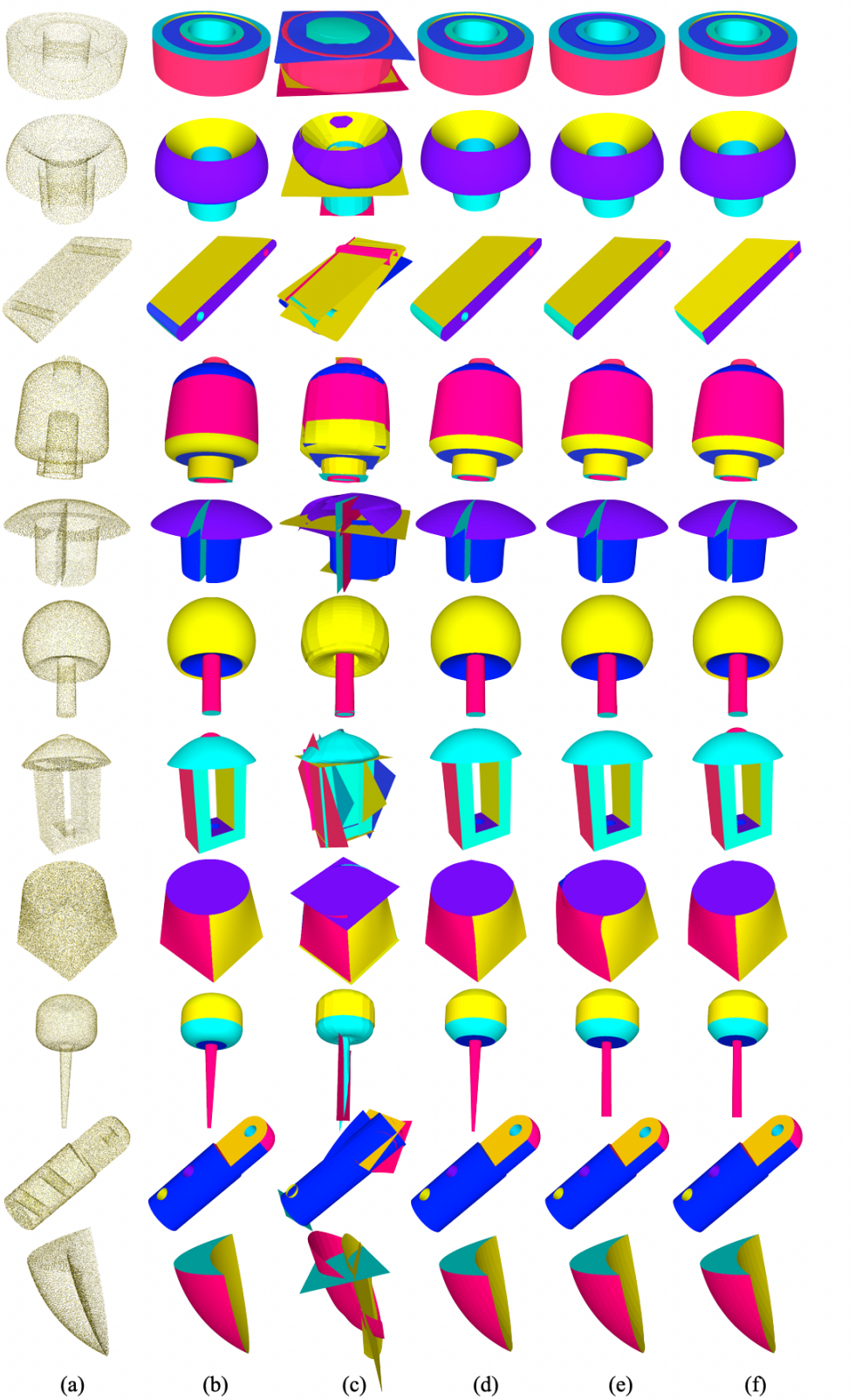

Figure 5. More visualisation results. From left to right: (a) input point cloud, (b) ground truth mesh, (c) reconstruction with ComplexGen, (d) Point2CAD with HP-Net, (e) Point2CAD with ParSeNet, (f) Point2CAD with GT segmentation.