Structured Semantic 3D Reconstruction – Solutions and Takeaways

Denys Rozumnyi
ETH Zurich
denys.rozumnyi@inf.ethz.ch

Abstract

This paper presents a winning solution to the Structured Semantic 3D Reconstruction Challenge at CVPR'24. A COLMAP- and geometry-based solution is proposed to extract approximately half of all vertices, which achieves high precision but low recall. An edge prediction procedure is also proposed, but it does not achieve better results due to the presented shortcomings of the used metric. These shortcomings lead to a very strong and simple solution, which can be written with one line of code. This one-line solution wins the challenge, and it is argued that it is very hard to beat with the current metric. Several ideas for improvement are proposed.

1. Introduction

The objective of the Structured Semantic 3D Reconstruction (S23DR) Challenge is to facilitate the development of methods for transforming posed images into a structured geometric representation (wire-frame), from which semantically meaningful measurements can be extracted [1].

The organizers provide a large dataset with 4316 samples for training and 175 for validation. The dataset contains various types of inputs, including COLMAP [3, 4] reconstruction, monocular depth, segmentations, and others. The ground truth is a graph with a set of vertices and edges between them. In the current form, the task is to predict such wire-frame reconstruction of mostly roofs of scanned houses.

2. COLMAP-based solution

The COLMAP-based solution is a geometry-based solution with no learning apart from hyper-parameters tuning on the training dataset.

2.1. Predicting vertices

The proposed solution takes as input the COLMAP point cloud and the gestalt segmentations. Then, we select all points that are suspected to be vertex points, which we later cluster with K-Means clustering. We repeat the same procedure for the apex and eave-end-point vertices separately using the same operations. Without loss of generality, we refer to a vertex label in the following explanations.

Point selection. In order to select relevant points from the COLMAP point cloud, each COLMAP point is projected to the image plane, and if it is a visible point in this view, we compute a distance to the closest vertex label (using a distance transform from a binary image of the given vertex label, either apex or eave-end-point). Then, we average this distance over all views where this point is visible. Finally, the points are selected if this distance is below the threshold Θ_{dist}, which is set empirically to 50 cm.

Density-based filtering. In order to filter outliers, we also remove points without enough support. We filter based on density, where we only keep points that have at least 3 neighbors within a distance of at most 40 cm.

K-Means clustering. The selected points that passed the density check are clustered by K-Means clustering (with termination criteria of 200 maximum iterations and threshold 0.3 cm). The correct number K of points is found in the following way. We start with $K = 1$ and iteratively increase it to at most $K = 30$ (maximum number of vertices in the training dataset). We iterate until adding another cluster would make the distance between the two closest clusters below a certain threshold Θ_{cluster}, set to 200 cm. Finally, the predicted vertices are all clusters that have at least 3 assigned points. This way, we generate our set of vertices (apex and eave-end-point separately).

2.2. Predicting edges

Given the estimated vertices, we iterate over all possible pairs of vertices and check whether it is an edge. First, we label apex vertices as $l_i = 1$ and eave-end-point vertices as $l_i = 0$. Then, for each possible edge (i, j), we compute their edge type as $l_i + l_j$. The type is 0 for gestalt label *eave*, 1 for *rake* and *valley*, and 2 for label *ridge*.

For each possible edge (i, j), we sample 20 points between them by linear interpolation and project them to each view, where both vertices i and j are visible. If there is no view where both vertices are visible, we discard such

edges. Then, for all sampled interpolated points, we project them to the image plane and again compute an average distance to the gestalt label given by the edge type (using morphological dilation and distance transform). Then, we keep only those edges for which the average distance is below a threshold Θ_{mean} and also, there is at least one view with an average distance below an even lower threshold Θ_{min} (meaning that there is at least one view where this edge is clearly visible). For each edge type, we use different thresholds. For type 0, it is 15 and 5 respectively, for both types 1 and 2, it is 40 and 20.

3. Understanding and fixing the metric

The most important part of the challenge is to understand every piece of the metric. The metric is called Wire-frame Edit Distance (WED), which is adopted from [2]. The metric contains a sum of 5 different terms:

- Vertex translation costs V_t.
- Vertex deletion costs V_d.
- Vertex insertion costs V_i.
- Edge deletion costs E_d.
- Edge insertion costs E_i.

After spotting several bugs in the metric, the metric has been fixed. However, it still needs a lot of improvement. In its current form, WED has several shortcomings:

- If the mean and standard deviation of the prediction and the ground truth vertices are very different, yet the solution looks relatively good (as in Fig. 2), the so-called *pre-registration* moves the predicted vertices to a wrong location. This is especially true when the prediction has some missing vertices.
- The dataset contains many vertices of small regions (as in Fig. 2 (right), which are never visible in the given inputs. Such vertices (and edges) are impossible to estimate. However, they change the mean/std of the prediction by a lot. This means that a high-quality reconstruction with such missing irrelevant vertices would have a high error due to this pre-registration.
- Penalizing missing vertices in the current form enforces the solution to predict more vertices, *e.g.* if we know that we always predict roughly half of the vertices, it is beneficial to add zero/dummy vertices (or better repeat the estimated vertices).
- Since pre-registration and bipartite matching change predictions by a wide margin, predicting edges is beneficial only in case almost all vertices are correctly estimated. A solution with all correct vertices but no edges would achieve WED of 1. This also means that if WED is much higher than 1, it does not make sense to predict edges at all. And given that the best-scoring solutions in this challenge were in the range of WED $= 2$, not dealing with edges was the best solution for everyone.
- Edge deletion E_d is computed over predicted vertices.

Method	WED
COLMAP-based (vertices, edges)	2.367
COLMAP-based (vertices)	2.352
COLMAP-based (vertices) + zeros	2.178
COLMAP-based (vertices) + center	2.057
COLMAP-based (vertices, edges) + repeat	1.963
one-line(18)	1.806
one-line(19)	1.760
one-line(20)	1.722
one-line(21)	1.689
one-line(22)	1.667
one-line(23)	1.657
one-line(24)	1.648
one-line(25)	**1.645**
one-line(26)	1.651
one-line(27)	1.659
one-line(28)	1.673
one-line(N_{pred})	1.706
one-line(N_{gt})	1.321

Table 1. **Ablation study of different methods.** We compare a COLMAP-based solution with a simple one-line baseline. The scores are averaged over 175 samples from the validation split.

One-line solution. All these shortcomings lead to one simple one-line solution, which achieves extremely low WED. All vertices are in the origin. The graph is fully connected. In the case that we know the ground truth number of vertices, this solution makes 4 out of 5 WED terms zero except for V_t. If the number of vertices is slightly off, we also have a vertex deletion or vertex insertion cost, but never both. The edge deletion is always zero, no matter how many vertices are predicted. On the other hand, the edge insertion is above zero if the predicted number of vertices is lower than the ground truth number of vertices. Therefore, it is better to overshoot than undershoot, *i.e.* predicting more vertices is better.

```
def one_line_solution(n):
    return np.zeros((n,3)), list(itertools.product(list(range(n)), list(range(n))))
```

Figure 1. **One-line solution.**

4. Experiments

We measure the scores on the provided validation set with 175 examples. All development and fine-tuning was done on the training dataset.

Ablation study. The main ablation study is shown in Fig. 1. The first five rows show different versions of predicting vertices. Empirically, we computed that our method produces around 45 % of ground truth vertices. Given the

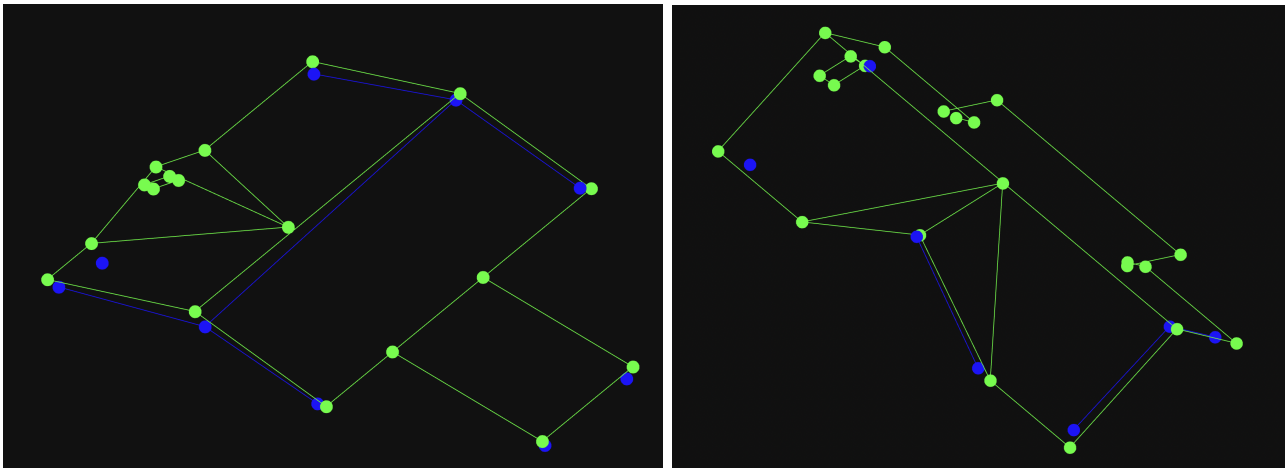

Figure 2. **Qualitative results.** Left: WED with edges is 1.773, without edges 1.661. Right: WED with edges is 2.206, without 2.073.

Method	WED	V_t	V_d	V_i	E_d	E_i
COLMAP-based (vertices, edges) + repeat	1.773	6202	458	0	2004	9187
COLMAP-based (vertices) + repeat	1.661	6202	458	0	0	10072
one-line($N_{\text{pred}} = 19$)	1.193	11559	458	0	0	0
one-line($N_{\text{gt}} = 18$)	1.148	11559	0	0	0	0
one-line(25)	1.467	11559	3212	0	0	0
one-line(17)	1.343	10543	0	917	0	2063

Table 2. **Metric terms** for a selected example from Fig. 2 (left).

shortcomings of the metric, we had to come up with a strategy to add more vertices (2.2 times more than we predict). Adding just points in origin already improves the score, adding a center of gravity of COLMAP points improves it even further. However, the best strategy is to augment vertices just by simply copying the same vertices one by one. As shown in rows 1 and 5, adding our edge prediction makes the performance worse. This is expected due to pre-registration, matching, and also those repeating vertices because it is no longer clear which one of them will be matched, and everything becomes very ambiguous.

The best score is achieved by one-line(25), which is above the average number of vertices in the dataset because it is better to predict more than less as already explained before. The one-line solution was discovered several hours before the deadline, and the submitted version is one-line(N_{pred}), where N_{pred} is the number of vertices predicted by the proposed method. However, at the time of writing this paper, it was found that one-line(25) achieves even better performance. An oracle-based method one-line(N_{gt}) achieves impressively high performance. Achieving this performance seems to be possible by training a regression model just to predict the number of vertices.

Qualitative results. Several examples are shown in Fig. 2. Our method has overall high precision but low recall. And such solutions are penalized a lot by the current metric. For instance, Fig. 2 (left) shows a good reconstruction. We predict 9 vertices, and with repetition, we have 19 vertices. Whereas the ground truth contains 18 vertices. In this case, our prediction with edges achieves WED of 1.773, and without edges 1.661. This is mostly due to the fact that after the *pre-registeration* of WED, the vertices are moved closer to the left side (where there are many irrelevant vertices that change the center of gravity), and then the bipartite matching is wrong. Every term of the metric for this example can be seen in Table 2.

5. Conclusion

We proposed several approaches to the wire-frame semantic 3D reconstruction task. Several ablation studies showed that the proposed one-line solution achieves very high performance. For future work, the metric needs to be adapted and the dataset cleaned (*e.g.* removing irrelevant wire-frames of small parts that are not even visible in the inputs). The metric could be improved by avoiding pre-registration. However, in our opinion, it must be completely re-designed. One possibility is to avoid measuring in 3D, but rather compare the re-projected wire-frames in 2D.

References

[1] Structured semantic 3d reconstruction (s23dr) challenge. In *CVPR*, 2024. 1

[2] Yujia Liu, Stefano D'aronco, Konrad Schindler, and Jan Dirk Wegner. Pc2wf: 3d wireframe reconstruction from raw point clouds. *ArXiv*, abs/2103.02766, 2021. 2

[3] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[4] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 1