Urban Reconstruction from Raw LiDAR Data with Subspace Modeling

Zhenxun Zhuang, 111131765
Stony Brook University
Stony Brook, NY, US
zzhuang@cs.stonybrook.edu

Abstract

Urban reconstruction is an active area of research with numerous applications. It deals with the recovery of 3D geometric models of urban scenes from various types of input data, among which LiDAR scanning is becoming increasingly popular. Despite its high accuracy, fast acquisition speed, and versatility, raw LiDAR data suffer from several major imperfections which hinder its prevalence in broader scopes. Those drawbacks include anisotropy of sampling density, contamination of noise and outliers, and missing large regions. Therefore, the commonly employed robust fitting techniques should be modified accordingly. Observing that the dominant objects in urban scenes are buildings which are composed of some primitive surfaces like planes and cylinders, the problem of local structure recovery can be modeled as substructure classification. In light of this, a novel framework is proposed to first automatically detect substructures around each point, and then combine these information to achieve the reconstruction of the whole scene. The results prove that this work is very promising.

1. Introduction

In recent years, urban reconstruction has attracted increasing attention in computer graphics, computer vision, and photogrammetry and remote sensing. Its goal is to reconstruct 3D geometric models from urban scenes, which consists of many objects such as buildings, streets, vegetation, and people [1]. It has found its way to many practical fields:

- Documenting the cultural heritage of our civilization.
- Rendering more realistic and vivid scenes of real cities appearing in movies and video games.
- Generating finer and more accurate digital maps.
- Forming the basis and references of urban planning.
- Offering a virtual platform for military exercises and real warfare.

However, the reconstruction problem is essentially ill-posed because there exists an infinite number of surfaces passing through or near the data points. Fortunately, we can overcome that by introducing priors: assumptions made by algorithms to tackle imperfections in the input data and recover as much information about the shape as possible [2]. Those assumptions can be imposed on the imperfections of the input data itself, on the scanned shape, or on knowledge of the acquisition. According to introduced priors, surface reconstruction algorithms can be classified into several categories: surface smoothness, visibility, volumetric smoothness, and so on.

In the context of urban reconstruction, scenes are mainly comprised of buildings, the surfaces of which are in turn composed of planes, cylinders, cones, and other primitive surfaces. Furthermore, as long as the desired level of detail is not too high, the surface of a building can be approximated and modeled using planes. Consequently, it is natural to make the prior that all substructures are planes. Of course, this prior can be extended to include any other parametric surface if required. The problem of local structure recovery can now be modeled as substructure classification.

The design of reconstruction algorithms relies heavily on the type of the input data. There are various types of input data which are suitable as a source for urban reconstruction, among which the LiDAR scanning is gaining more and more popularity.

LiDAR is the acronym for Light Detection And Ranging, an optical remote sensing technology that finds range and/or other information of a distant target by measuring properties of scattered light. Current ranging laser systems mainly use pulse lasers, in which the range to an object is determined by measuring the time delay between transmission of a pulse and detection of the reflected signal [3].

As suggested by its name, the difference between LiDAR and RADAR (RAdio Detection And Ranging) is that RADAR uses radio waves while LiDAR uses light rays, or the lasers to be more precise. Compared with radio waves used in RADAR, lasers possess attractive properties such as high beam density, excellent coherency and parallelism,

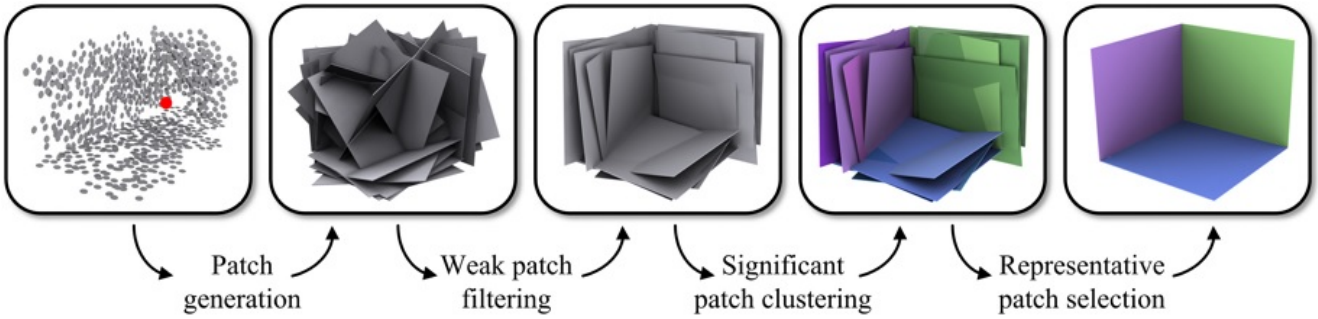


Figure 1: The framework of the proposed subspace modeling algorithm.

much smaller wavelengths, and closeness to monochromatism. These attributes enable LiDAR to detect small objects invisible at RADAR frequencies since it is only possible to image an object about the same size as, or larger than, the wavelength. The very narrow beam width nature of lasers also allows the mapping of physical features with very high resolution. However, raw LiDAR data has some special characteristics that make its processing very challenging:

Density Anisotropy The point density of LiDAR scan varies according to the measuring distance, the angle between the scanned surface of current object and the scanning direction, as well as the shape's geometric features. As a rule of thumb, the nearer the distance or the smaller the slope, the denser the scan data. This imperfection directly strikes the very foundation of most existing data classification or clustering methods in that they are based upon the basic assumption that samples are uniformly distributed on each subspace. Only from the assumed density isotropy does evaluating substructures by counting their corresponding inliers appear reasonable.

Noise and Outliers Inevitably, raw LiDAR data are contaminated by noises and outliers. A special case is that in scenes with multi-structures, even the inliers of one substructure will be considered as outliers of another.

Unknown Number of Substructures No prior knowledge is available when it comes to determine the number of substructures; the number must be automatically detected.

To this end, Wang and Xu proposed a novel subspace modeling framework for shape detection and reconstruction from raw LiDAR data [4]. To deal with density anisotropy, a density-independent metric was devised to measure the goodness of a patch. To handle noise, non-parametric kernel density estimate techniques combined with scale estimation was introduced. To determine the unknown number of substructures, a patch similarity metric based on residual preference permutations was developed. This proposed algorithm will be developed in Section 2.

2. Scene Recovery using Subspace Modeling

The framework of the subspace modeling algorithm proposed in [4] is illustrated in Figure 1. For the local neighborhood of each point, a set of initial surface patches are generated (Section 2.1), from which the weak ones are filtered out (Section 2.2). The remaining significant patches are then clustered into several groups (Section 2.3). Finally, representative patches are extracted from each group to represent the underlying substructures (Section 2.4). Based on this, we can achieve scene recovery by globally clustering points with similar representative patch sets and refitting a model for each cluster (Section 2.5).

2.1. Initial Patch Generation

In order to successfully recover the local structure, the initial patch set must be large enough to cover all underlying substructures. RANSAC, the acronym for RANdom SAmple Consensus, is employed to generate initial patches [5]. It works in the following manner:

Given a model M which requires m points to be uniquely instantiated, and a point set P whose size n is larger than m.

1. Randomly select m points from P to get an instance of M, denoted by M_1.

2. For any point in P, if it locates within a certain error tolerance ϵ to M_1, accept it into the consensus set S_1.

3. If the size of S_1 is larger than a pre-defined threshold t, use all points in S_1 to compute a new instance of M via the least squares method, and keep it as an initial patch; otherwise, discard M_1.

4. Repeat step 1-3 until enough patches are generated.

2.2. Significant Patch Determination

The initial patch set includes both good and weak patches in the sense that how well they approximate underlying substructures. Therefore, it is essential to filter

out weak ones. We achieve this by assigning each patch a score denoting its goodness, and then discarding all patches whose scores are not high enough. Ideally, if the inliers of a patch come from one single substructure, its score should be as high as possible; otherwise, its score should be close to zero. After assigning each patch a score, a threshold is needed to dichotomize good patches and week ones.

2.2.1 Noise Scale Estimation

The accuracy of scale estimation significantly affects the performance of many robust estimators such as RANSAC, Hough Transform, and J-linkage. With an accurate scale, we can dichotomize inliers and outliers, select the best hypothesis, or determine various data dependent parameters. In this project, I adopted the Iterative Kth Ordered Scale Estimator proposed in [6].

Given a model instance M_1 and a point set P whose size is n, we can obtain the residual set $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$ which measures the distance from each point to the model instance M_1. Denote the sorted absolute residual set by $\{|\tilde{r}_i|\}$ such that $|\tilde{r}_1| \leq |\tilde{r}_2| \leq \ldots \leq |\tilde{r}_n|$.

With the scale of inlier noise s, outliers can be separated from inliers using the following equation:

$$|r_i/s| < \mathrm{E} \tag{1}$$

where E is a threshold related to the assumed noise distribution. For a Gaussian distribution, a value of 2.5 can ensure that 98 percent of inliers will be included.

Alternatively, from a given inlier set, noise scale can be estimated using the Kth Ordered Scale Estimator (KOSE):

$$\hat{s}_K := |\tilde{r}_K|/\boldsymbol{\theta}^{-1}(\frac{1}{2}(1+\kappa)) \tag{2}$$

where $\boldsymbol{\theta}^{-1}(\cdot)$ is the argument of the normal cumulative density function, and $\kappa(\kappa := K/n)$ is the fraction of the whole data points with absolute residuals less or equal to the Kth ordered absolute residual.

Here one is facing the "chicken or the egg" dilemma because the precision of the scale estimation depends on accurately dichotomizing inliers and outliers, and vice versa.

Therefore, IKOSE is proposed to iteratively optimize the estimate of the size of the inlier set, and is described in Algorithm 1.

The most important issue in IKOSE is the choice of the K value. A higher K value can include more inliers which will lead to better statistical efficiency, but this will potentially cause the algorithm to break down especially when the outlier level is high because the Kth ordered point will possibly be an outlier. Since it is more important to avoid breakdown, a small K value is recommended. We set it to 10% of the number of neighbors of a point, and do not change it for any individual task.

Algorithm 1: $\hat{s}_K = \mathbf{IKOSE}(\mathbf{r}, \mathrm{K})$

Input : a residual set $\mathbf{r}$ with size n and the K value.
Sort $\mathbf{r}$ into $\{|\tilde{r}_i|\}$.
Set $\hat{n}_0 = n$ and $\hat{\kappa}_0 = \mathrm{K}/n$.
while $|\hat{s}_{K,t} - \hat{s}_{K,t-1}| \geq 10^{-6}$ and $\hat{\kappa}_t < 1$ **do**
  Estimate the scale $\hat{s}_{K,t+1}$ by (2).
  Update the inlier set by (1).
  Calculate $\hat{n}_{t+1}$ and $\hat{\kappa}_{t+1}$.
end
Output: the scale estimate $\hat{s}_K (= \hat{s}_{K,t})$.

2.2.2 Evaluating Patches

The weighing metric is based on nonparametric kernel density estimate techniques.

For an arbitrary patch $\boldsymbol{\theta}$, its variable bandwidth kernel density estimate at r is:

$$\hat{f}_{\boldsymbol{\theta}}(r) := \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\hat{h}(\boldsymbol{\theta})}\mathbf{KN}\left(\frac{r - r_i(\boldsymbol{\theta})}{\hat{h}(\boldsymbol{\theta})}\right) \tag{3}$$

where $\mathbf{KN}(\cdot)$ is the kernel and $\hat{h}(\boldsymbol{\theta})$ the bandwidth.

The Epanechnikov kernel $\mathbf{KN}_E(r)$ is used which is defined as:

$$\begin{aligned}\mathbf{KN}_E(r) &:= \begin{cases}\frac{3}{4}(1-||r||^2) & ||r|| \leq 1\\ 0 & ||r|| > 1\end{cases}\\ \mathbf{kn}_E(r) &:= \begin{cases}1-r & 0 \leq r \leq 1\\ 0 & r > 1\end{cases}\end{aligned} \tag{4}$$

where $\mathbf{kn}_E(r)$ is the profile of the kernel $\mathbf{KN}_E(r)$.

The bandwidth can be estimated using:

$$\hat{h}(\boldsymbol{\theta}) = \left[\frac{243\int_{-1}^{1}\mathbf{KN}(r)^2 dr}{35n\int_{-1}^{1}r^2\mathbf{KN}(r)dr}\right]^{\frac{1}{5}}\hat{s}_K(\boldsymbol{\theta}) \tag{5}$$

where $\hat{s}_K(\boldsymbol{\theta})$ is the noise scale of $\boldsymbol{\theta}$ estimated using IKOSE.

In order to suppress patches with large noise scales, a patch is scored by:

$$\hat{w} := \frac{\hat{f}_{\boldsymbol{\theta}}(\mathbf{O})}{\hat{s}_K(\boldsymbol{\theta})} = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{KN}(r_i(\boldsymbol{\theta})/\hat{h}(\boldsymbol{\theta}))}{\hat{s}_K(\boldsymbol{\theta})\hat{h}(\boldsymbol{\theta})} \tag{6}$$

If the noise scale of each patch is similar, then the bandwidth is also similar. Thus, (6) can be rewritten as:

$$\hat{w} \propto \frac{1}{n}\sum_{i=1}^{n}\mathbf{KN}(r_i(\boldsymbol{\theta})/\hat{h}) \tag{7}$$

Notice that this is just a weighted count of the number of points falling within a certain fixed range of the patch.

Since the above weighting metric measures the goodness of a patch solely by its number of inliers, it works well when the assumption of sampling density isotropy holds true. However, it fails to handle the situation when the input is raw LiDAR data in which the sampling density of points varies significantly.

Therefore, the weighting metric is refined by introducing a sampling-independent area-based measure based on the observation that the goodness of a patch should relate to the size of its underlying substructure. Specifically, for an arbitrary patch $\boldsymbol{\theta}$, first construct the k Nearest Neighbor graph from its inlier set and find the largest connected component χ. The weighting metric (6) is then modified as:

$$\hat{w} := \frac{\hat{f}_{\boldsymbol{\theta}}(\mathbf{O})}{\hat{s}_K(\boldsymbol{\theta})} \cdot \sqrt{|\chi|} \cdot \sum_{e_i \in E} |e_i| \tag{8}$$

where $|\chi|$ is the size of χ, E the set of edges in χ, and $|e_i|$ the length of e_i.

2.2.3 Threshold Determination

After assigning each patch a score, weak patches can now be filtered out. Instead of manually set a cut-off value, a data-driven thresholding method is employed.

Suppose a score set $W = \{w_1, w_2, \ldots, w_m\}$ has been obtained in the last step. We first scale the scores within the range of $[0, 1]$ by $w_i = \frac{w_i - w_{min}}{w_{max} - w_{min}}$ and then calculate the normalized histogram of the scaled score set. Assuming the histogram has l columns, denote the weight value of the jth column in the histogram as v_j, and its corresponding frequency as f_j. Let v_j be the threshold, the probability mass functions of the weak and significant patches can be defined as:

$$\begin{aligned} \mathcal{P}_w(v_j) &= \sum_{0 \le k \le j} f_k \\ \mathcal{P}_s(v_j) &= \sum_{j < k \le l} f_k \end{aligned} \tag{9}$$

and the corresponding entropies:

$$\begin{aligned} H_w(v_j) &= -\ln \sum_{0 \le k \le j} \left[\frac{f_k}{\mathcal{P}_w(v_j)} \right]^2 \\ H_s(v_j) &= -\ln \sum_{j < k \le l} \left[\frac{f_k}{\mathcal{P}_s(v_j)} \right]^2 \end{aligned} \tag{10}$$

The optimal threshold is the one that maximizes the overall entropy:

$$\hat{w} := \arg\max_j H(v_j) = \arg\max_j (H_w(v_j) + H_s(v_j)) \tag{11}$$

It can be determined by exhaustion since the size of histogram is relatively small.

With the optimal threshold, one can now dichotomize significant patches and weak ones. Those with a score less than the optimal threshold are considered weak, and discarded.

2.3. Clustering

In the retained significant patch set, a substructure usually has more than one corresponding patches. Thus, it is necessary to cluster all significant patches into several groups each representing a distinct underlying substructure.

In the context of clustering significant patches, the distance measure should describe the similarity of two patches in the sense of how close they are with respect to the same underlying substructure. Here we introduce two distance measures.

The Jaccard Distance Denote the inlier sets of two patches $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ as $\mathcal{C}(\boldsymbol{\theta}_i)$ and $\mathcal{C}(\boldsymbol{\theta}_j)$, respectively. The Jaccard distance between these two patches are defined as:

$$\mathcal{J}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) := 1 - \frac{\mathcal{C}(\boldsymbol{\theta}_i) \cap \mathcal{C}(\boldsymbol{\theta}_j)}{\mathcal{C}(\boldsymbol{\theta}_i) \cup \mathcal{C}(\boldsymbol{\theta}_j)} \tag{12}$$

Obviously, the more $\mathcal{C}(\boldsymbol{\theta}_i)$ and $\mathcal{C}(\boldsymbol{\theta}_j)$ overlap, the smaller the distance.

The Kendall's tau Distance While the Jaccard distance measures the similarity between the elements of two sets, the Kendall's tau distance further measures the similarity of the *order* of elements in two sets [7]. It calculates the number of pairs (x, y) which satisfy:

$$\begin{gathered} ((|r_{i,x}| < |r_{i,y}|) \wedge (|r_{j,x}| \ge |r_{j,y}|)) \\ \text{or} \\ ((|r_{i,x}| > |r_{i,y}|) \wedge (|r_{j,x}| \le |r_{j,y}|)) \end{gathered} \tag{13}$$

where $|r_{i,x}|$ is the absolute residual of a point x with respect to a patch $\boldsymbol{\theta}_i$.

Observing that the preferences of inliers from the same substructure towards a set of patches are correlated, the Kendall's tau distance is chosen.

Assuming there are n points and m significant patches, we can compute the residual set $\mathbf{r}(\boldsymbol{\theta}_i)$, where $i = 1, 2, \ldots, m$.

Sort $\mathbf{r}(\boldsymbol{\theta}_i)$ into increasing order, denoted by:

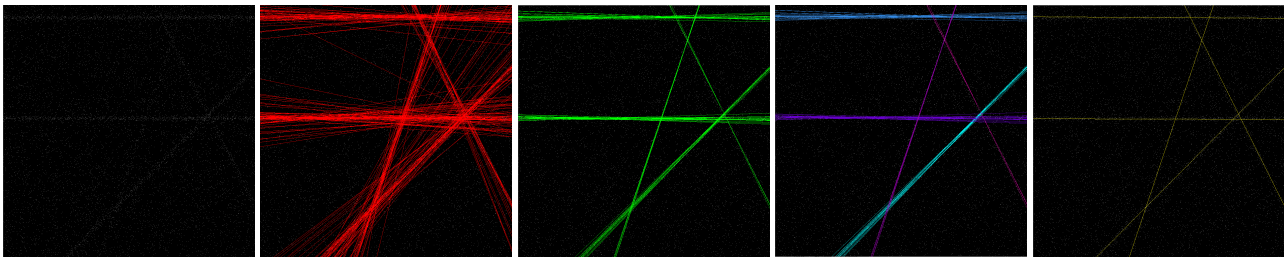
$$\tilde{\mathbf{r}}(\boldsymbol{\theta}_i) = [r_{i,\mu_1^i}, r_{i,\mu_2^i}, \ldots, r_{i,\mu_n^i}] \tag{14}$$

Thus, the preference permutation of all points to $\boldsymbol{\theta}_i$ is defined as:

$$\boldsymbol{\pi}(\boldsymbol{\theta}_i) = [\mu_1^i, \mu_2^i, \ldots, \mu_n^i] \tag{15}$$

Denote $\mathbf{R} = \{\boldsymbol{\pi}(\boldsymbol{\theta}_1), \boldsymbol{\pi}(\boldsymbol{\theta}_2), \ldots, \boldsymbol{\pi}(\boldsymbol{\theta}_m)\}$.

Since the number of substructures is unknown, the medoid shift algorithm [8] is employed in the original paper



(a) Point cloud (b) Initial patches (c) Significant patches (d) Clusters (e) Representative patches

Figure 2: Pipeline of substructure modeling in 2D, there are in total 10000 points, and the noise level is 2% while the outlier rate is 70% (better viewed in electronic edition due to the high density of points).

to conduct non-parametric clustering. It is a mode-seeking algorithm that can compute shifts toward areas of greater data density using local weighted medoids, and can automatically determine the number of clusters during execution.

However, I noticed that the Kendall's tau Distance is already good enough to find similar representative patches. Two patches from different substructures always have a much larger distance (usually around 0.5) than that between two from a same substructure (usually less than 0.05). Moreover, it turns out that the medoid shift algorithm tends to oversegment the significant set. Therefore, I directly use the Kendall's tau Distance to cluster significant patches, and it works very well.

2.4. Representative Patch Extraction

From each cluster, the patch with the highest score are selected as the representative patch of that cluster. As a result, the local structure around a point can now be described by the representative patch set.

2.5. Scene Recovery

Since the neighborhood of each point can be characterized by its underlying substructures, which in turn can be described by corresponding representative patch set, we can cluster all points based on the similarity between their representative patch sets, thus separate the input point cloud to several meaningful clusters. The following metric is introduced to measure this similarity:

$$sim(p,q) := \frac{|\mathcal{G}(p) \cap \mathcal{G}(q)|}{max(|\mathcal{G}(p)|, |\mathcal{G}(q)|)} \tag{16}$$

where $\mathcal{G}(p)$ is the representative patch set of point p, and $|\mathcal{G}(p)|$ is the size of $\mathcal{G}(p)$, i.e., the number of its elements. Therefore, $|\mathcal{G}(p) \cap \mathcal{G}(q)|$ means the number of similar representative patches between the neighborhood of point p and q. The similarity of two patches is measured by the difference between their parameters.

Based on the observation that the denser the points, the better the results of subspace modeling, we set the size of neighborhood automatically according to the size of the whole point cloud to ensure that the subspace modeling procedure will have enough data to accurately derive representative patches. Moreover, when a point has too few neighbors, it will be considered as an outlier, and won't go through following clustering steps.

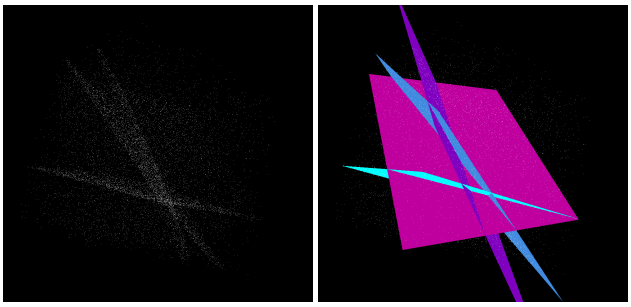
After obtaining representative patch sets for the neighborhood around each point, the region growing strategy is employed to find similar points. Specifically, we first randomly choose an unclustered point as the seed, then search its neighbor for similar points and add them into the cluster. The searching will proceed recursively on neighborhoods of newly added points following the Depth First Search manner. The process will end until no more non-outlier points can pass the similarity test, and one cluster is got. This clustering procedure will be repeated iteratively until all non-outlier points have been assigned to its cluster.

The next step is to recompute models that fit each cluster. Here we calculate a plane using all points in one cluster via least squares method to represent its underlying substructure. The final thing is to determine the boundary of each plane. This can be achieved by finding the bounding box of each plane, i.e., the range in which all points fall in. All fitted planes now constitute the whole reconstructured scene.

3. Experiments

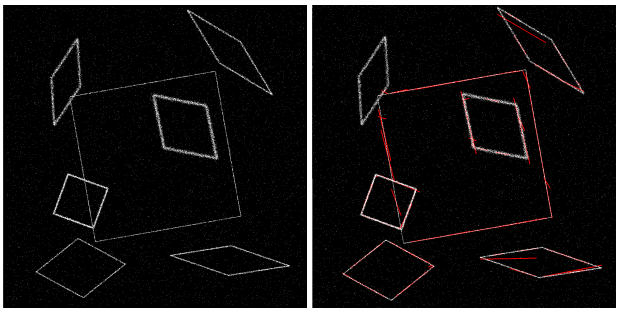
Substructure Modeling Figure 2 shows the result for 2D and Figure 3 for 3D. The input point cloud is synthesized using a Matlab program written by myself, which is able to specify the size, line parameters, density of each line, noise level, and outlier percentage of point clouds. As can be seen, the performance is pretty good that all lines or planes are detected even with a relatively high level of outliers and noise.

Scene recovery The results for 2D are shown in Figure 4, whose input point cloud is generated using Matlab. Most



(a) Point cloud (b) Representative patches

Figure 3: Results for substructure modeling in 3D, there are in total 20000 points, and the noise level is 2% while the outlier rate is 30% (intermediate results are not shown since too many planes leads to a very cluttered view).

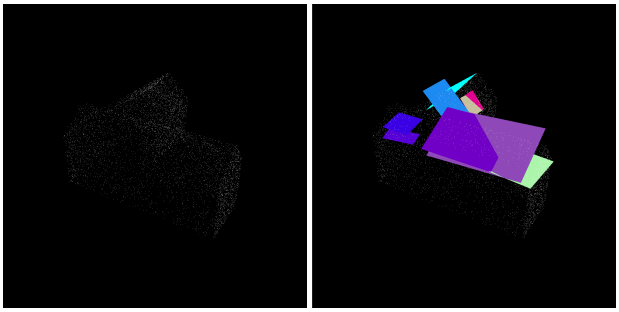


(a) Point cloud (b) Reconstructured scene

Figure 4: Results for scene reconstruction in 2D, there are in total 30000 points (you may need to look closely since some of red lines are covered by white points, especially in those regions of high point density).

lines are successfully detected though there exist some errors. In the experiment, I observed that the more points the input contains, the better the results. This also holds true when it comes to individual structures that a structure with a larger density is more likely to be recovered. In addition, I also noticed that a big structure is usually much easier to detect than a small one. Since our model only uses one line to fit each cluster, it cannot process corners and intersections of lines. Also, the bounding strategy can be improved by introducing techniques like O-Snap [9].

Figure 5 shows our results on 3D real world data. Since the dataset is too small, the performance is really bad. However, with a denser input, our model can achieve better results.



(a) Point cloud (b) Reconstructured scene

Figure 5: Results for scene reconstruction in 3D, there are in total 5035 points (better viewed using my interactive demo which can allow you to move, rotate, and scroll).

4. Conclusion

In this project, I implemented an algorithm which tries to reconstruct urban scenes from raw LiDAR data using local substructure classification. The greatest merit of this technique is that it is fully automatic, free of any user choice of parameters. It makes three major contributions: the effective noise scale estimation, a sampling-independent area-based weighting metric, and the residual preferences permutation.

Nevertheless, this technique is far from the ultimate solution that can automatically recover all structures in a scene with high precision. Actually, there is no such solution because a scene always consists of infinite structures of different scales. Rather, the technique is only able to fit all "significant" structures that has enough inliers and are large enough to be detected.

References

[1] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Gool, and W. Purgathofer, "A survey of urban reconstruction," in *Computer graphics forum*, vol. 32, pp. 146–177, Wiley Online Library, 2013.

[2] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the art in surface reconstruction from point clouds," in *EUROGRAPHICS star reports*, vol. 1, pp. 161–185, 2014.

[3] A. Wehr and U. Lohr, "Airborne laser scanningan introduction and overview," *ISPRS Journal of photogrammetry and remote sensing*, vol. 54, no. 2, pp. 68–82, 1999.

[4] J. Wang and K. kai Xu, "Shape detection from raw lidar data with subspace modeling," *IEEE Transactions on Visualization and Computer Graphics*, 2016.

[5] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[6] H. Wang, T.-J. Chin, and D. Suter, "Simultaneously fitting and segmenting multiple-structure data with outliers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 6, pp. 1177–1192, 2012.

[7] M. G. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[8] Y. A. Sheikh, E. A. Khan, and T. Kanade, “Mode-seeking by medoidshifts,” in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.

[9] M. Arikan, M. Schwärzler, S. Flöry, M. Wimmer, and S. Maierhofer, “O-snap: Optimization-based snapping for modeling architecture,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, p. 6, 2013.