Surface Reconstruction based on Self-Merging Octree with Deep Learning

Jian Lv WOAIWODEXUEXI@GMAIL.COM
Xie Han HX14317@163.COM
Jiajie Zheng JESSEZHENG112233@PROTONMAIL.COM
Fengguang Xiong HOPENXFG@NUC.EDU.CN
Min Pang PMIN@NUC.EDU.CN
School of Data Science and Technology, North University of China, Taiyuan, 030051, China

Editors: Wee Sun Lee and Taiji Suzuki

Abstract

A model segment method called Octree Subdivision has been presented for long years, which allows any three-dimensional point cloud object to be subdivided into infinitesimals so that it can be approximated by a particular surface function. In this paper, we proposed a new method named self-merging octree to reconstruct the surface of 3D Point Cloud which can be obtained by laser scanners or generated by some 3D modeling software. Different from any other surface reconstruction algorithms such as local property-based or specific type-based, a function pool-based was introduced in our research because it can express many different types of surfaces. We subdivide point cloud model by self-merging octree and categorize it by the neuro-network. In this idea, it is easy for us to find a proper surface function to present the subsurface of the model. What‘s more, while we extend the function pool, we can indicate far more style models. We have tried to reconstruct many point cloud models‘ surfaces in this way, and it works well and also shows its potential ability to build a bridge in the fields of model editing, model splicing, and model deformation.

Keywords: point cloud; octree; deep learning; surface reconstruction

1. Introduction

Surface reconstruction, also called reverse engineering or model re-edit, is a significant assignment in industrial design and manufacture. Nowadays, there are many more CAD, CAM and CAE softwares providing real-time design and modification, visualization, digitization, etc. to help people efficiently and effectively produce some very interesting and colorful objects. But this still relies on the designer‘s intuition and empirical experience to rebuild a model quickly and rightly. That means you need to spend a lot of efforts learning how to use modeling softwares, and getting sufficient design experience in constant experimentation when you want to become a good designer and design amazing productions. With the booming demand for film animation, aircraft steamship, cultural relic protection, unmanned driving, virtual reality, artificial intelligence, and other industries, the three-dimensional reconstruction task has risen to a new height. This requires not only accurate and realistic design models but also time limits and reconfigurability. In order to solve these problems, many researchers have done countless work to release people from complex and boring reconstruction work. Thanks to the rapid development of computer technology, scientists from over the world have developed a variety of model representation. At present, there are several regulation approaches for drawing curves such as NURBS curve, Hermite curve, Bezier curve, B-spline curve, etc. In addition, many mechanisms of 3D surface drawing have been invented, such as Coons surface, Bezier surface, B-spline surface, derived T-spline, G-spline, rational Gaussian surface, as well as generalized rational parameter surface. Those curve and surface rendering algorithms have long been widely used, making a significant contribution to industrial design and manufacturing.

Different from the digital and virtual world which is defined by discrete numeric value and showed by limit pixels on the screen, the real physical world is subdivided continuously and infinitely. So it is impossible for us to describe the real world around us as detailed as what we can see by computers. The more subtle ways we use to regenerate the virtual world, the more complex data we will get. Meanwhile, it is not intuitive and uneasy for us to deal with that. There is a new data structure named 3D Point Cloud which can reproduce the real world simply, robustly, and intuitively. Due to the irregularity of point cloud data, it is not easy for us to use it directly for 3D reconstruction. The question of how to rebuild the real world based on point cloud has become a hot topic in current research.

Machine learning is a branch of artificial intelligence, including the acknowledgment of deep learning and neural networks. As early as 1956, the concept of artificial intelligence (AI) was first proposed by McCarthy at the Dartmouth Conference. He tried to use the machine to simulate human intelligence which we are born with. With the advances in internet, artificial intelligence and Big Data, machine learning is widely used to deal with the issue of image processing, object recognition, data mining, search engines, medical imaging and object generation. At present, the commonly used machine learning frameworks mainly include DNN called deep neural network, RNN called cyclic neural network, GAN called the generation of confrontation networks, CNN called convolutional neural networks, etc. Recently, some scholars have applied these techniques to manifold surface reconstruction and established an emerging perspective to understand machine learning from a geometric perspective. Some researches show that machine learning relies on the manifold distribution theorem, which indicates that there is a large application space in surface reconstruction with the technology of machine learning.

To gain a higher efficiency of surface reconstruction and obtain a more extensive application, we promote a new measure combining all of the above techniques and exploring further. Our research is based on basic point cloud data, and we use octree algorithm that has combined with a machine learning method. In this work, we chose one named PointNet Qi et al. (2016) which can directly perform point cloud partitioning and output each class label for the whole input. Unlike the segment method of traditional octree and Adaptive-Octree Wang et al. (2018a), we create a novel self-merging octree algorithm which will help us look for the most suitable partition location. With the application of this method, it is not only easy for us to avoid unnecessary surface stitching that may be represented by an equation or an intact surface at the end of the program of the surface conflated but also it can also make the reconstructed surface smoother and more natural. The establishment of the equation pool includes so many samples of the simple surface represented formulation such as planar and Bezier surface and so on. So while we input more types of the surface, we can represent more surface and decrease much time to making margination.

2. Related Work

Surface construction is a hot research field in recent years since the methodology of surface stimulation has been raised last century. In 1988, David and Richard proposed a hierarchical subdivision technique based on B-spline, which can change control vertices and local characteristics of the presentation of the surface. In 1995, Grimm et al. proposed the manifold modeling Grimm and Hughes (1995) of arbitrary topological surfaces based on extended B-spline functions, inheriting the characteristics of traditional B-spline local control and continuity. In 2000, Navau et al. proposed a surface modeling Navau and Garcia (2000) of arbitrary topological meshes based on tensor product B-spline functions. Based on the two manifolds with arbitrary topological structure, a continuous surface model with C^k was constructed. In 2002, Cindy came up with a parametric simple surface modeling Grimm (2002) which was modeled by simple parameterization of the given surface. In 2003, John et al. constructed a parametric hyperbolic polygon of the N hole Tori and generated the surface by a standard spline function Grimm and Hughes (2003). In 2004, Zorin et al. proposed an arbitrary smooth surface reconstruction Ying and Zorin (2004) based on a simple manifold, satisfying linear dependence at the control vertices, and the constructed surface has C^∞ continuously. In 2005, Gu et al. proposed a manifold spline technique to construct a spline surface Gu et al. (2005) with high parameters and no need to be repaired by concentrating a triangular B-spline surface from a planar domain to multiple domains, thus completing any geometric shape and topology modeling well. In 2006, He proposed a manifold T-spline technique He et al. (2006) to construct a high-quality and controllable layered structure through global conformal parameterization. In 2007, Wang et al. proposed a polyhedral spline technique Wang et al. (2007), which considers polygons to have rectangles everywhere in their domains. Except for a few corners, the boundaries of polyhedra can be naturally decomposed into a set of regular structures. In 2008, He et al. used the mathematical theory of Ricci flow to construct a manifold spline function with a single singular point and gave strict proof to it Gu et al. (2008), which eased the modeling difficulty of manifold splines and promoted a wide range of manifolds in entity modeling. In 2009, Vecchia et al. constructed a manifold surface with sharp features using the principles of manifold structure and hybrid approximation Vecchia and Jüttler (2009). In 2011, Elif et al. combined the methods of Ying and Zorin to construct a manifold surface with boundaries Tosun and Zorin (2011). In 2012, Li et al. discussed the linear independence of the T-spline mixing function and determined the function matrix of the T-spline to NURBS transformation Li et al. (2012). Subsequently, some people have used the form such as sparse expression method Wang et al. (2016), triangle B-spline function Gu et al. (2008), Powell-Sabin spline (vertex interpolation and normal), T-spline function, polyhedron The T-spline Liu et al. (2015), local feature retention spline, multi-level triangular PSP spline Li and Tian (2015) and other techniques to complete the surface reconstruction work, and also achieved good results.

These methods and measures have their own advantages and disadvantages, but when dealing with singular points or edges, sharp points, and other irregular features, the processing algorithm is slightly complicated and insufficient. In response to these problems, recently, Zhang et al. used polygon mesh technology to construct manifold surfaces Zhang and Liu (2017). Through the combination with indeterminate basis functions, the constructed surface can perfectly represent the local information of the object, such as sharp features, smoothness, etc. In this way, Surfaces expressed by basis functions can be used to approximate real objects. Quan et al. collect RGB photos directly through the camera, splicing the photos to directly form a stereo effect, as a result the resulting three- dimensional scene not only satisfies the requirements of virtual modeling but also acquires the model with real Texture Quan et al. (2015); Wang et al. (2018b). The data sources for these photos are usually acquired using drones. Due to the good freedom and high-speed mobile performance of drones, they are favored by researchers nowadays. At the same time, Luo et al. introduced a 3D laser scanner to scan existing objects directly and collect 3D point cloud information of objects Luo et al. (2016), especially for large-scale scenes, which makes modeling more convenient. These point cloud information collected from the scanner can be processed directly as input data or to generate grids for further operation. Because the point cloud information is too large, and unpredictable errors such as noise and holes will occur during the acquisition process, it is difficult for the computer to quickly and conveniently construct models by using the existing algorithm without performing the necessary preprocessing. There is a number of feasible processing algorithms as to the removal of point cloud noise Wang and Cho (2015) and hole repair Wang et al. (2018b). By using these existing algorithms to preprocess point cloud data, relatively clean point cloud data can be obtained, which can simplify subsequent processing in the next step.

Therefore, the work of surface reconstruction gradually tends to be based directly on point cloud data. Our work is also based on the above, and we combine machine learning methods to make the modeling process more unsupervised. Furthermore, we offer another possibility for modeling methods based on the point cloud data, which will make modeling easier, when we provide enough surface samples that can be represented by functions. At the same time, when all the surfaces of our surface pool adopt the design rules in computer-aided design software, we will allow the machine to automatically construct the existing models in the existing world based on point cloud data, which will greatly simplify the design. The design process of the personnel also provides a convenient method for redesigning the real world model.

3. Construction

In this section, we will introduce the algorithm for surface construction via self-merging octree. For a given point cloud data, we first surround it with a bounding sphere and then create an external cube. In this way, we can use octree algorithm to segment this point cloud into a number of a small piece of point clouds which occupies the same amount of spaces from its cube bounding box. With a series of information packaged of the above pieces of the point cloud, we can use this packaged information of the subspace as raw input to machine learning. Through the discrimination of neural network in machine learning, the type to which each piece of point cloud belongs can be determined. Based on these results, we can indicate specific surface representations to them. Eventually, using surface stitching technology, surface construction of the entire model is completed. And the pseudo code can be described as follows.

Algorithm 1 The workflow framework of our system.

```
Input: The point cloud, $S$;
Output: The fitting surface of point cloud $S$, $H$;
  Step 1: Constructing a bounding box for point cloud $S$; Initializing the iteration depth $\theta = n$, and the number of iterations $k = 0$;
  Step 2: $k = k + 1$; Performing an octree partitioning algorithm on the input point cloud set $S$ to obtain a point cloud subset, $S_i$;
  Step 3: if $S_i$ is not an empty point set, Packing it as the input to the classifier to get the type of each point cloud, $\Gamma_t$;
  Step 4: if $S_i$ has not been marked with any type, go to step2; else if $S_i$ has been marked with the type $\Gamma_t$, go to step5;
  Step 5: Finding the spatial domain node $S_j$ of $S_i$. if $S_j$ and $S_i$ are the same types, $\Gamma_t$, merging them as $S_{ij}$; if the type of $S_{ij}$ is also $\Gamma_t$, recording the merge information in variable $A$, otherwise, cancel this merger; if $k \geq n$, or when all $S_i$ have been typed with tags, go to step6, otherwise go to step2;
  Step 6: According to the type of $S_i$ to construct a corresponding fitting surface, and use the fusion algorithm to splicing and merging adjacent surfaces. Finally, the fitted surface $H$ of the model $S$ is obtained;
  return $H$;
```

3.1. Self-merging Octree

The method of octree Meagher (1982) was proposed by Meagher in 1982, which can progressively divide the three-dimensional space into eight octants by using the iterative algorithm. Fig. 1 has shown one segmentation molder of the traditional octree method. Our octree method also works on this foundation. But it is different from the traditional octree algorithm which has to set the recursion depth θ for the stopping of the program, our method can stop itself automatically without setting this parameter. However, it is a wise choice for us to initialize the depth θ to ensure the stability of our program.

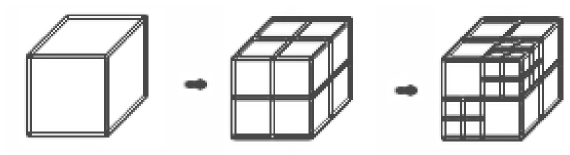
Figure 1: Traditional octree segmentation model.

When a point cloud data S is given, we first create a bounding sphere to surround the point cloud and then create an external bounding box that can be rotated around the sphere. In order to make point cloud data segmentation easier to process by our program, we randomly select some bounding boxes for comparison and choose the best one. Normally, adjusting the 3D point cloud data to the axis-aligned mode will reduce time for this step. Since we identified a cube bounding box, we began a one to eight subdivision of the point cloud as the traditional way do. Thus, we can get eight subspaces $s_i (i = 0, 1, ..., 7)$ from every piece of the point cloud to be segmented, each subspace has a similar spatial size. Then, we suspend the segmentation processing and start to judge whether s_i is empty. If it is empty, the slice space will be ignored. while it is not empty, it will be packaged as input data that can be processed by our point cloud classifier which can classify these inputs into existing types based on our surface pool. To simplify the complexity of the program, the feasibility of the experimentation and the less cost of the time of the experimental verification, we assume that our classifier can only perform two sorting operations — plane or other. However, Our classifier is so powerful that it can classify a wide variety of patch types while we have trained it with various types of surfaces first. The classification diagram is shown as Fig. 2.

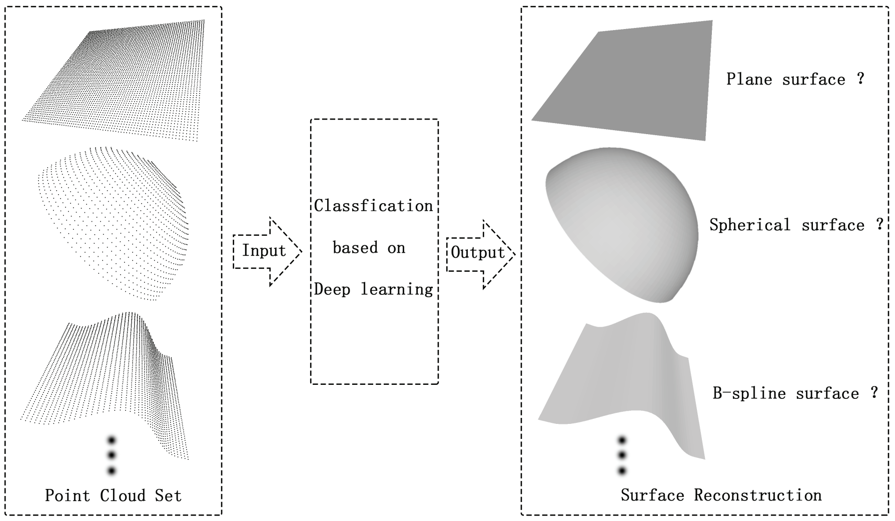

Figure 2: Classfication model.

Once the classifier (Section 3.2) has determined the category of each input piece of the point cloud and outputs the labels of every input, the self-merging operation will begin automatically. According to the data structure of the octree, each piece point cloud can easily find its neighbors. If the point cloud and its neighbor are of the same type and do not belong to other types, then the two pieces point cloud will be merged as the new piece point cloud $p_j(j = 0, 1, ..., 17)$ which will be taken as the input to the classifier after all merge operation is finished. While all of the steps of the self-merging process is over, the classification operation will be activated again to label the new input. If the merged data does not belong to any type of the pre-merger, the merge will be canceled, otherwise, it will be regarded as valuable. In order not to destroy the data structure of the octree algorithm, we will save these point cloud data in the form of files and create a new data structure which is similar to the octree structure to store the merge information of each node of the octree. After an iteration, we will get four types of point cloud data from the input s_i, they are an empty point set s_m, an unmerged point set s_n with available labels, a merged point set p_k with the available label and a set of points s_l with other labels. For these types of point clouds, we will handle differently in the next iteration. the point set s_m which is empty will be ignored and the point set s_l which has not been identified type will need to be subdivided by the program in the next iteration and do the same operation as the first iteration. Meanwhile, the point set s_n and p_k which are marked with the label will only need to participate in the merge operation of the next iteration. We will repeat those steps until all the piece of the subpoint cloud has been classified.

To ensure normal operation of the program, we initialized the iteration depth value θ based on experience for the different model. We all know that three points can form a plane, and any line and single point can be considered as planes. Therefore, when our θ value is empty that can be regarded as infinite, the program can also run normally and stop by itself automatically instead of splitting to each point. Sometimes, when our point cloud data contains some noise. We can eliminate the few faces of these point sets which only contain very few points, which can optimize the effect of reconstruction. More, we can calculate the average normal vector of each patch of the surface to determine if it is noise-induced and consider whether to ignore this part of the content. Of course, this will make the sharp features of the model smooth or even disappear.

Following this theory, we only need to extend the capabilities of the classifier to enhance the program‘s ability to express surfaces. For example, we can add Gaussian surfaces, Spline surfaces and some surfaces with sharp features Wang et al. (2016) to our surface pool, which allows our program to fit surfaces with distinctive features. In order to display the visualization of the operation of our method better, We tried our algorithm on a two-dimensional curve, which is shown in Fig. 3. For a given curve model S, we assume that it can be fitted with some spline that can be expressed by simple equations. Our goal is to find these straight segments and merge them into the simplest straight segments. In Fig. 3, we use the gray area to represent the straight line segments that have merged, and those large white space areas are the space areas we have discarded. As the subdivision is deeper, we can get the desired result we expect. It can be seen from the experimental results that our algorithm works out, and this is not the same as the adaptive O-CNN algorithm Wang et al. (2018a) which is based on convolutional neural networks for the voxelization of the model. Therefore, by implementing a piecewise linear fit, all curves can be well characterized.

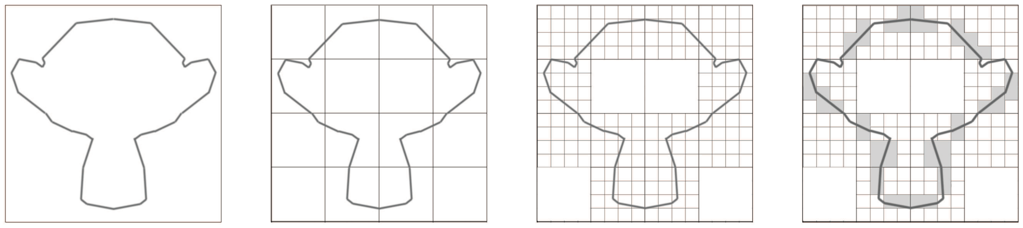

Figure 3: Application of self-merging algorithm on curves.

3.2. Classifier

Using the octree segmentation algorithm of the above, we can get some spatial subdivision of point clouds. The significant work to do is to determine the type of each point cloud based on the surface pool, which will promote our program on performance well. In order to process the data better to obtain a good classification effect, we introduce the PointNet++ Qi et al. (2017) processing theory into our machine learning network, which can directly handle point clouds that are always disordered, irregular and anomalous sets. As Qi said, for a given point cloud data $\{x_1, x_2, ..., x_n\}$ where $x_i \in R$. we can define a mapping function $f : x \rightarrow R$ as shown in Eq.1 to process point cloud data Qi et al. (2016). The γ and h in the Eq.1 are MLP (multi-layer perceptron) networks.

$$f(x_1, x_2, ..., x_3) = \gamma(\max_{i=1,...,n} \{h(x_i)\}) \tag{1}$$

According to the mathematics description, we pack each piece of the segmented point cloud as the data input of the neural network and calculate the type tag of the point cloud as the output result of the classier. Its operation process is shown in Fig. 4.

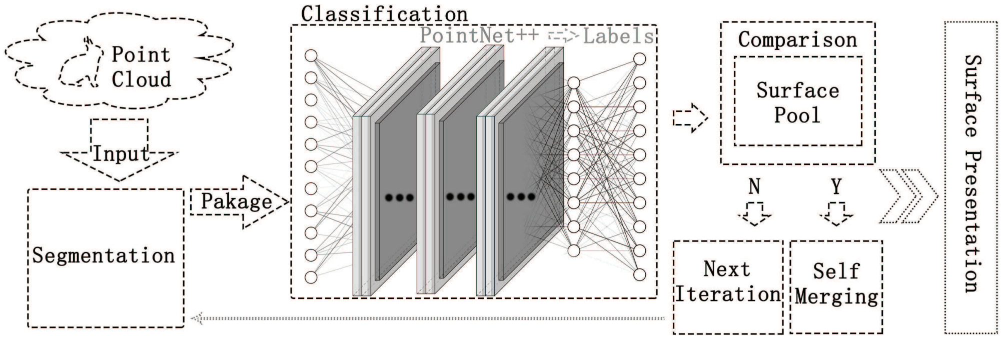

Figure 4: Processing of the point cloud.

Different from the usual classification methods, our requirements for classification accuracy can be dynamically adjusted based on the classification capabilities of the classifier. When our classifier has a strong classification ability, which means there are many types of patches of the point cloud that can be distinguished. We only need to improve the classification accuracy requirements so that each point cloud can be tagged with the corresponding label. When our classification ability is weak, for example, it can only distinguish between planar and non-planar types, we should have no more stringent accuracy requirements for classification, in case the subdivision is too deep and the running time is too long. By adjusting the classification accuracy dynamically, the program is more robust, which can make classification results more available basically. At the same time, we do not need to encode and decode the point cloud data to find the most optimized feature at the end of merging together, which leaves our classification results not constrained by the ability to the representation of the feature extraction. Our purpose is only to get the classification label to inform the segment handler whether to perform the next iteration.

3.3. Surface Presentation

When our classification and self-merging process completed all the operations, we began to perform surface reconstruction operations based on well-typed pieces of point cloud data.

Assuming that all surfaces in our surface pool can be represented by using a specific cubic B-spline surface, it is easy to know that for different types of neighbors domain surfaces, only spline interpolation algorithms are needed to fuse them together. As pointed out by Liu, a $m \times n$ secondary B-spline surface $S(u, v)$ can be defined as Eq.2 in the space with the number of $(m + 1) \times (n + 1)$ control points Liu et al. (2018) which can be set as $p_{i,j}(i = 0, 1, ..., m; j = 0, 1, 2, ..., n)$. The $N_{i,m}$ and $N_{j,v}$ are the basic function of the B-spline in the Eq.2, and where $(u, v) \in [0, 1] \times [0, 1]$.

$$S(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{i,j} N_{i,m}(u) N_{j,n}(v) \tag{2}$$

Since there are some calculation errors in the segmentation process, we leave a certain overlap for each segmented region which can refer to the paper Zhang and Liu (2017). In our testing, we expanded the length of the stereo box edge of each divided convolutional 1.1 times. We can think that the splicing operation of the two patches is to fuse the overlapping regions by using three-dimensional stitching technology. And then using the Laplacian smoothing algorithm, the smoothing operation is performed on each merged portion. Suppose that the surface we reconstructed last is denoted by H, and the finite q surfaces in the surface pool are represented by $h_i(i = 0, 1, ..., q-1)$ which can be represented by known parametric equations. And the each-patch surface is represented by $g_j(j = 0, 1, ..., n)$. The surface construction of the entire model S can be expressed by the following formulation.

$$H = \sum h_i + \sum g_j \tag{3}$$

It is known from the above formulation. For the better surface reconstruction of the model, we only need to optimize the surface expressions in the surface pool and choose a more sophisticated stitching algorithm.

4. Results

Similar to other surface construction methods, we use the method of this paper to perform surface fitting on point cloud data. First, we tested the feasibility of our approach on a set of regular data sets, some of which were primarily downloaded from the Stanford University web page. The experimental results are shown in Fig. 5. The experimental results show that our visualization results are similar to those of the existing excellent algorithms which can reconstruct surface fitted with the given point cloud data very well. We overlay the reconstructed surface and the point cloud to better demonstrate the reconstruction results of our algorithm. For a better explanation, we give a visual comparison chart with Zhang‘s algorithm Zhang (2017). It can be seen from the figures that our algorithm has obvious advantages in the visualization effect because it can fit the point cloud model well.

And then we added some Gauss noise to the data sets to test the reliability of our program, the result is shown in Fig. 6.$a1-a3$. The $a1$ is the original point cloud, $a2$ is the point cloud with Gauss noise addition, and $a3$ is the test result of our method. At the same time, we also tested the algorithm of Zhang Zhang (2017) on it, unfortunately, the program crashed directly. It can be seen from the experimental results that our method is

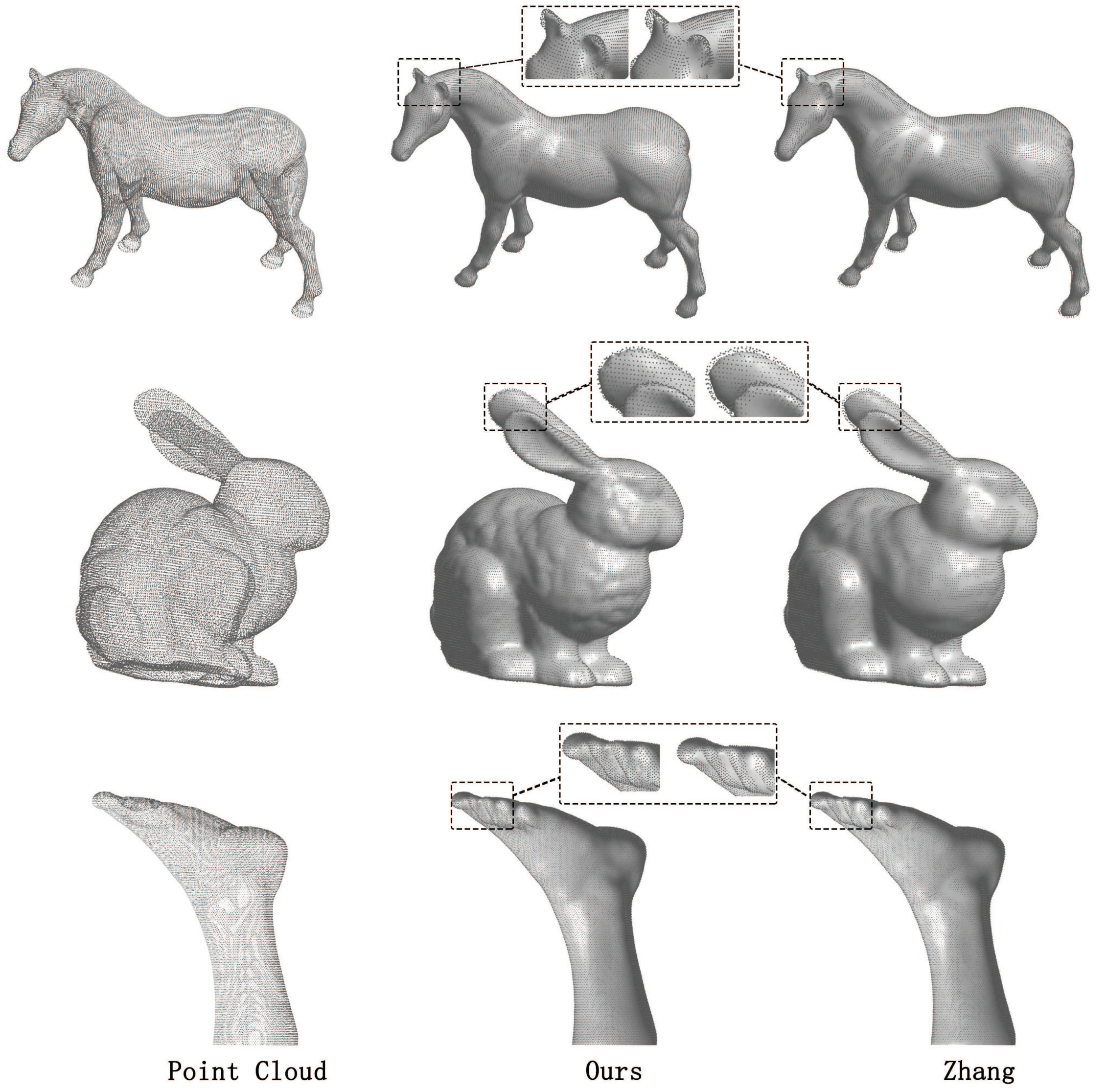

Figure 5: Results and comparison.

sensitive to noise, but it can barely reconstruct similar results. This also means that our method also requires some pre-processing to eliminate noise for better results. Finally, We used the point cloud data scanned by the laser scanner to test our method. The test results are shown in Fig. 6.*b*. It can be seen from the experimental results that our method is also very robust to the point cloud data obtained on the real site.

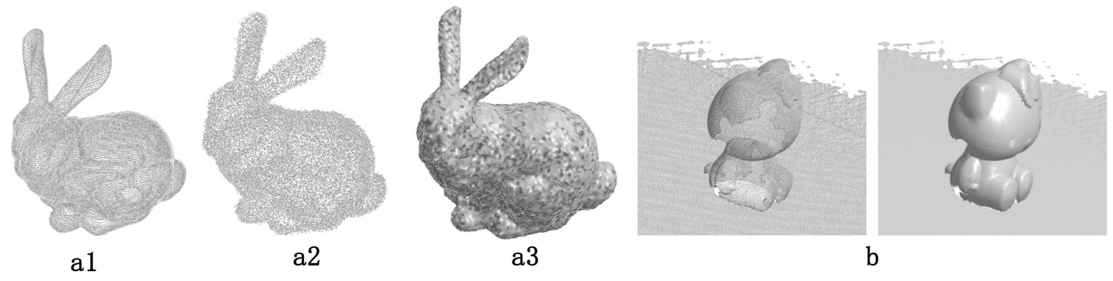

Figure 6: $a1-a3$: Test result on point cloud with Gauss noise; b: Test results on a point cloud collected by a laser scanner.

5. Conclusions and Limitations

A new method named self-merging octree has been proposed in this paper, which has tried to construct the surface of the three-dimensional model successfully. We also combine with the machine learning in our classier process progress. At the end of the last of the program, cubic B-spile interpolation has been used to combine all of the species of the point cloud, which works well in our data set. In this way, we can easily extend the type of pieces surface in our surface pool. The more types we own, the more abilities our program can perform to the surface of the model. But in the real world, the point cloud collected by scanner usually has noise and hole. The model with some noise would mislead our classier to make the wrong decision and the hole of the model would be misfit by our combination functions. Those problems need us to do a preprocess first to remove noise and fit the hole effectively. On the other hand, due to the complex process steps, our method always takes much time to operate once especially for the irregular model which contains some sharp features and singularities. In the future, we will try to bring in some other type surfaces to enlarge our surface pool to enhance the ability of our program. We will also follow Wang‘s theory Wang et al. (2017) to reduce the time complexity by using GPU parallel computing. Meanwhile, for a better effect of surface stitching in our mathematics, we will introduce the manifold principle and bring in some geometric theories to machine learning Lei et al. (2018), which would fit well with our self-merging octree algorithm. Furthermore, The sparse sample method is also worth trying, and it has been successfully used in the manifold surface reconstruction algorithm of polygon mesh Zhang and Liu (2017).

Acknowledgments

In our work, we sincerely thank the seniors and juniors who provided detailed experimental procedures and open source code, as well as the students and teachers who gave us a lot of help. At the same time, our work has been supported by the National Natural Science Foundation of China (No. 61672473, 61379080), which allows us to go further and more motivated on our experimental schedule. At the same time, we are also very grateful to all the reviewers for their valuable advice and selfless guidance.

References

Cindy Grimm. Simple manifolds for surface modeling and parameterization. In *2002 International Conference on Shape Modeling and Applications (SMI 2002), 17-22 May 2002, Banff, Alberta, Canada*, pages 237–246, 2002. doi: 10.1109/SMI.2002.1003551. URL https://doi.org/10.1109/SMI.2002.1003551.

Cindy Grimm and John F. Hughes. Modeling surfaces of arbitrary topology using manifolds. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995, Los Angeles, CA, USA, August 6-11, 1995*, pages 359–368, 1995. doi: 10.1145/218380.218475. URL https://doi.org/10.1145/218380.218475.

Cindy Grimm and John F. Hughes. Parameterizing n-holed tori. In *Mathematics of Surfaces, 10th IMA International Conference, Leeds, UK, September 15-17, 2003, Proceedings*, pages 14–29, 2003. doi: 10.1007/978-3-540-39422-8_2. URL https://doi.org/10.1007/978-3-540-39422-8_2.

Xianfeng Gu, Ying He, and Hong Qin. Manifold splines. In *Proceedings of the Tenth ACM Symposium on Solid and Physical Modeling 2005, Cambridge, Massachusetts, USA, June 13-15, 2005*, pages 27–38, 2005. doi: 10.1145/1060244.1060249. URL https://doi.org/10.1145/1060244.1060249.

Xianfeng Gu, Ying He, Miao Jin, Feng Luo, Hong Qin, and Shing-Tung Yau. Manifold splines with a single extraordinary point. *Computer-Aided Design*, 40(6):676–690, 2008. doi: 10.1016/j.cad.2008.01.008. URL https://doi.org/10.1016/j.cad.2008.01.008.

Ying He, Kexiang Wang, Hongyu Wang, Xianfeng Gu, and Hong Qin. Manifold t-spline. In *Geometric Modeling and Processing - GMP 2006, 4th International Conference, Pittsburgh, PA, USA, July 26-28, 2006, Proceedings*, pages 409–422, 2006. doi: 10.1007/11802914_29. URL https://doi.org/10.1007/11802914_29.

Na Lei, Zhongxuan Luo, Shing-Tung Yau, and Xianfeng David Gu. Geometric understanding of deep learning. *CoRR*, abs/1805.10451, 2018. URL http://arxiv.org/abs/1805.10451.

Qingde Li and Jie Tian. Multilevel refinable triangular psp-splines (tri-psps). *Computers & Mathematics with Applications*, 70(8):1781–1798, 2015. doi: 10.1016/j.camwa.2015.07.017. URL https://doi.org/10.1016/j.camwa.2015.07.017.

Xin Li, Jianmin Zheng, Thomas W. Sederberg, Thomas J. R. Hughes, and Michael A. Scott. On linear independence of t-spline blending functions. *Computer Aided Geometric Design*, 29(1):63–76, 2012. doi: 10.1016/j.cagd.2011.08.005. URL https://doi.org/10.1016/j.cagd.2011.08.005.

Lei Liu, Yongjie Zhang, Yang Liu, and Wenping Wang. Feature-preserving t-mesh construction using skeleton-based polycubes. *Computer-Aided Design*, 58:162–172, 2015. doi: 10.1016/j.cad.2014.08.020. URL https://doi.org/10.1016/j.cad.2014.08.020.

Shuyu Liu, Xie Han, and Caiqin Jia. Cubic b-spline-interpolation based mesh splicing and fusion. *Journal of Image and Graphics of China.*, 23(12):1901–1909, 2018. URL http://en.cnki.com.cn/Article_en/CJFDTotal-ZGTB201812011.htm.

Huan Luo, Cheng Wang, Chenglu Wen, Zhipeng Cai, Ziyi Chen, Hanyun Wang, Yongtao Yu, and Jonathan Li. Patch-based semantic labeling of road scene using colorized mobile lidar point clouds. *IEEE Trans. Intelligent Transportation Systems*, 17(5):1286–1297, 2016. doi: 10.1109/TITS.2015.2499196. URL https://doi.org/10.1109/TITS.2015.2499196.

Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(1):85, 1982. doi: 10.1016/0146-664X(82)90128-9. URL https://doi.org/10.1016/0146-664X(82)90128-9.

Josep Cotrina Navau and Núria Pla Garcia. Modeling surfaces from meshes of arbitrary topology. *Computer Aided Geometric Design*, 17(7):643–671, 2000. doi: 10.1016/S0167-8396(00)00020-0. URL https://doi.org/10.1016/S0167-8396(00)00020-0.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL http://arxiv.org/abs/1612.00593.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5105–5114, 2017. URL http://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature-learning-on-point-sets-in-a-metric-space.

Long Quan, Jianxiong Xiao, Tian Fang, and Peng Zhao. Generating three-dimensional façade models from images, 2015. URL http://patentsearch2.ttc.ust.hk/ListPatents.asp?ttccode=TTC.PA.407&search=Search.

Elif Tosun and Denis Zorin. Manifold-based surfaces with boundaries. *Computer Aided Geometric Design*, 28(1):1–22, 2011. doi: 10.1016/j.cagd.2010.07.005. URL https://doi.org/10.1016/j.cagd.2010.07.005.

Giovanni Della Vecchia and Bert Jüttler. Piecewise rational manifold surfaces with sharp features. In *Mathematics of Surfaces XIII, 13th IMA International Conference, York, UK, September 7-9, 2009, Proceedings*, pages 90–105, 2009. doi: 10.1007/978-3-642-03596-8_6. URL https://doi.org/10.1007/978-3-642-03596-8_6.

Chao Wang and Yong K. Cho. Smart scanning and near real-time 3d surface modeling of dynamic construction equipment from a point cloud. *Automation in Construction*, 49:239 – 249, 2015. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2014.06.003. URL http://www.sciencedirect.com/science/article/pii/S0926580514001381. 30th ISARC Special Issue.

Hongyu Wang, Ying He, Xin Li, Xianfeng Gu, and Hong Qin. Polycube splines. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling, Beijing, China, June 4-6, 2007*, pages 241–251, 2007. doi: 10.1145/1236246.1236281. URL https://doi.org/10.1145/1236246.1236281.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: octree-based convolutional neural networks for 3d shape analysis. *ACM Trans. Graph.*, 36 (4):72:1–72:11, 2017. doi: 10.1145/3072959.3073608. URL https://doi.org/10.1145/3072959.3073608.

Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive O-CNN: a patch-based deep representation of 3d shapes. *ACM Trans. Graph.*, 37(6):217:1–217:11, 2018a. doi: 10.1145/3272127.3275050. URL https://doi.org/10.1145/3272127.3275050.

Ruimin Wang, Ligang Liu, Zhouwang Yang, Kang Wang, Wen Shan, Jiansong Deng, and Falai Chen. Construction of manifolds via compatible sparse representations. *ACM Trans. Graph.*, 35(2):14:1–14:10, 2016. doi: 10.1145/2835488. URL https://doi.org/10.1145/2835488.

Xiaochao Wang, Jianping Hu, Dongbo Zhang, Lixin Guo, Hong Qin, and Aimin Hao. Multi-scale geometry detail recovery on surfaces via empirical mode decomposition. *Computers & Graphics*, 70:118–127, 2018b. doi: 10.1016/j.cag.2017.07.024. URL https://doi.org/10.1016/j.cag.2017.07.024.

Lexing Ying and Denis Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Trans. Graph.*, 23(3):271–275, 2004. doi: 10.1145/1015706.1015714. URL https://doi.org/10.1145/1015706.1015714.

Chun Zhang and Ligang Liu. Manifold construction over polyhedral mesh. *Communications in Mathematics and Statistics*, 5(3):317–333, Sep 2017. ISSN 2194-671X. doi: 10.1007/s40304-017-0113-x. URL https://doi.org/10.1007/s40304-017-0113-x.

Dandan Zhang. Integral surface reconstruction method based on manifolds. *North University of China.*, 23:1–77, 2017. URL http://new.oversea.cnki.net/KCMS/detail/detail.aspx?dbcode=CMFD&dbname=CMFD201702&filename=1017167230.nh.