AN ABSTRACT OF THE THESIS OF

Zehuan Chen for the degree of Master of Science in Computer Science presented on January 10, 2019.

Title: Web-based Deep Segmentation of Building Structure

Abstract approved: ______________________________

Fuxin Li

Deep learning and neural network has been widely used in research, deep learning has empowered many tasks such as point clouds segmentation and shape recognition. One of the main advantages of deep interaction point cloud segmentation is that it allows the feature extraction can be learned through neural network based on a large amount of dataset. Our focus is large point clouds, we propose a variety of measuring tools to analyze and validate raw point cloud data, which is the web-based deep segmentation user interaction on large point clouds. It allows users to view data sets with millions of points, from sources such as building structure and indoor scene, in standard web browsers, and processing 3D point clouds deep segmentation with the neural network. The interaction tools can assist to distinguish building structure and non-building structure in one room.

Web-based Deep Segmentation of Building Structure

by

Zehuan Chen

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented January 10, 2019
Commencement June 2019

ACKNOWLEDGEMENTS

I would like to thanks who has contributed to my work and this thesis. To my advisor, Fuxin Li, for his help with this thesis and his patient guidance of my research path. To Wenxuan Wu as well as Erzhuo Che for their previous works, on which this thesis is based. Thanks to Michael James Olsen, Associate Professor and his colleagues of Civil and Construction Engineering school, for making their amazing contributions to providing and collecting a huge amount of point clouds data. Special thanks to family and friends, who encourage me all the time during the time I pursuing my MS degree.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

1.1 Motivation

Point clouds are three-dimensional (3D) models that consist of points in space area, differ from other 3D model representations, such as mesh, volumetric model and depth map. Point clouds are more commonly obtained by directly scanning the real world through various scanning methods, such as laser scanning and photogrammetry. Point cloud data is close to raw sensor data and can transfer to other 3D model representations and products by further processing, modeling, and analysis.

Scan data can be used in many applications, such as tracking construction progress monitoring and analysis of the building structure, generation of 3D models, and motions of games or movies (e.g. Dynamic capturing technique). In addition, point clouds are also widely used in real-time perception, such as lidar (light detection and ranging) perception in the autonomous driving vehicle and robotic SLAM (simultaneous localization and mapping). For most of these applications, point clouds are usually treated as raw data. These point clouds require to be further processed such that useful information can be extracted.

Building Information Modeling (BIM) is a process involving the generation and management of digital representations called "Building information models" (BIMs), which are files which can be extracted, exchanged or networked to support decision-making regarding a building or other built asset. It is also enabling its users to share and compare their models seamlessly throughout the whole lifecycle of their project, from planning to demolition. It offers a variety of benefits including prevention of data loss from one phase to another, better visualization and coordination between different trades involved in a project. Thus, BIM implementation in the construction industry has increased exponentially, especially in the last decade.

Scan-to-BIM, which is a process of 3D laser scanning a physical space or site, to create an accurate digital representation of the scanning. This representation can be used for designing progress or evaluating options. The data collection part in Scan-to-BIM helps to remove a large degree of human error from the surveying process and enables a significantly high volume of data to be collected over a considerably shorter period of time. Then, in order to build BIM environment, the second part of Scan-to-BIM calibrate scanning data to share with others or import the data into the project's common data environment (CDE) and enables to analyze, visualize and model from the scan. This end-to-end approach can deliver users more information about their designed proposals, scope of the work or verify the progress of the work.

Most existing features for point cloud are used to apply handcrafted towards specific tasks. Point features often encode certain statistical properties of points and are designed to be invariant to certain transformations, which are used to classify as intrinsic or extrinsic. They can also be categorized as local features and global features. For a specific task, it is not trivial to find the optimal feature combination. To find the optimal feature combination, Deep learning can provide the ability to approach the optimal feature extraction by learning from existed point cloud data and task result, and could significantly reduce data complexity for modeling and data analysis in an automated Scan-to-BIM framework.

With the release of big data distribution, WebGL, and server framework over web browsers has become increasingly popular. All those related techniques are evolved into a standard and can be supported by most of main-stream browsers. It enable the developers, artists, companies, researchers are willing to share their content with a wide audience and access their work on any device in any place without the need to install additional software.

Furthermore, the widespread use of the neural network demand the high-performance calculation resources, multi-core CPU and GPU for example, especially when user requires working remotely without these powerful resources. Web-based communication architecture allows users to work with massive 3D data and deep learning algorithm regardless of using any device that with or without the high-performance calculation ability. Many services, like Sketchfab[17], Potree[14] emerged that allow users to upload, share and view content without any knowledge about the underlying 3D modeling and geometry mechanics.

This paper introduces a prototype of web-based end-to-end point cloud processing toolbox that is capable of data loading, viewing, neural network semantic scene segmentation and point cloud manual annotation. This work will improve and provide support to a complete Scan-to-BIM framework.

1.2 Problem Definition

3D scanning technologies such as laser scanners or photogrammetry produce enormous amounts of data. Due to the nature of point data, it is easy to directly obtain the coordinate data through scanning method, however, the raw data directly from the scan are lack of refined data type such as normal or texture that people need to post process it through some specific calculations method.

Common two-dimensional interaction metaphors (e.g. mouse) are useful tools when selecting or picking regions from a two-dimensional context. When porting these techniques to 3D, the third dimension (i.e. depth) must be guessed or controlled separately. A region of interest is usually a set of points that are spatially neighboring and create a structural element in the point cloud. Ideally, the selection of such a region is performed by defining a minimal enclosing volume that contains all points. Achieving this selection by using 2D-interaction metaphors only is challenging, as the techniques do not know the desired depth boundaries of the selection region. Therefore, interactions across multiple views are needed to achieve this selection.

1.3 Structure of the work

Chapter 2 gives an overview of related work, including other viewers for large point clouds, and work related to high-quality rendering of point clouds. The powerful features of deep neural network that we been applied on our system is given an introduction with details in Chapter 3. Chapter 4 describes how a web-based point cloud interaction is worked. This includes the calculation of colors and point sizes, as well as generation of nicely uniformly distributed colormap, which computes illumination and outlines with only the input of class numbers. Chapter 4 covers the implementation details of Javascript and WebGL, and describes some of the functionality in our system. Performance evaluations and some of our own results are shown in Chapter 5. A conclusion and a non-exhaustive list of future tasks is given in Chapter 6.

Chapter 2: Related Work

2.1 Web-based Massive Point Cloud rendering

Potree[15], a web-based renderer for large point clouds. It allows users to view data sets with billions of points, from sources such as LIDAR or photogrammetry, in real time in standard web browsers. One of the main advantages of point cloud visualization in web browser is that it allows users to share their data sets with clients or the public without the need to install third-party applications and transfer huge amounts of data in advance. The focus on large point clouds, and a variety of measuring tools, also allows users to use Potree to look at, analyze and validate raw point cloud data, without the need for a time-intensive and potentially costly meshing step. The streaming and rendering of billions of points in web browsers, without the need to load large amounts of data in advance, is achieved with a hierarchical structure that stores subsamples of the original data at different resolutions. A low resolution is stored in the root node and with each level, the resolution gradually increases. The structure allows Potree to cull regions of the point cloud that are outside the view frustum, and to render distant regions at a lower level of detail. The result is an open source point cloud viewer, which was able to render point cloud data sets of up to 597 billion points, roughly 1.6 terabytes after compression, in real time in a web browser.

PointCloudViz server[19] and the corresponding web client are a commercial service by Mirage Technologies and a complement to their free desktop LIDAR viewer. Their system uses a multiresolution structure for efficient streaming and rendering. Notable features include oriented splats, lighting, different materials such as RGB, intensity and classification and the modification of the color gradiants for elevation and intensity through sliders.

ShareLIDAR[16] is a multi-resolution point cloud renderer with hosting service. Notable features include illumination through normals, an orthographic top view, a section (height-profile) tool and the adjustment of point sizes to reduce holes. A downside is that it loads data in smaller tiles that do not cover the whole data set. This leads to large empty space while the user waits for the data to stream in.

2.2 Construction segmentation algorithm

The classification and segmentation algorithm is a core operation module in our system, the paper of Che et al, "Fast edge Detection and Segmentation of Terrestrial LASER Scans through Normal Variation Analysis"[2], introducing a novel way of construction segmentation algorithm. Terrestrial Laser Scanning (TLS) utilizes light detection and ranging (lidar) to effectively and efficiently acquire point cloud data for a wide variety of applications. Segmentation is a common procedure of post-processing to group the point cloud into a number of clusters to simplify the data for the sequential modelling and analysis needed for most applications. This paper presents a novel method to rapidly segment TLS data based on edge detection and region growing. First, by computing the projected incidence angles and performing the normal variation analysis, the silhouette edges and intersection edges are separated from the smooth surfaces. Then a modified region growing algorithm groups the points lying on the same smooth surface. The proposed method efficiently exploits the gridded scan pattern utilized during acquisition of TLS data from most sensors and takes advantage of parallel programming to process approximately 1 million points per second. Moreover, the proposed segmentation does not require estimation of the normal at each point, which limits the errors in normal estimation propagating to segmentation. Both an indoor and outdoor scene are used for an experiment to demonstrate and discuss the effectiveness and robustness of the proposed segmentation method.

Chapter 3: Deep Learning on point sets applications

Deep learning on 2D images has been vastly researched in the past few years. It achieves excellent results on segmentation, classification, and other special tasks, it needs to thanks two main things: convolutional neural networks and tons of image data. For three dimensional world, the data is now growing rapidly. Whether it originated from a human designed CAD model or a scanned point cloud from a LiDAR sensor or an RGB-D camera, point clouds can be obtained in everywhere and the requirement of getting more 3D point clouds data is inevitable.

Therefore, in our research, the system can have high compatibility to apply these amazing deep learning tools on.

3.1 Handcrafted point cloud features without deep learning

Before the time that people use deep learning so frequently as today, people processing 3D related task mainly through traditional point cloud features. Most existing features for point set data type are customized towards specific tasks. Point features often encode certain statistical properties of points, this design of invariant to certain transformations usually classified as intrinsic or extrinsic, Or sometimes people also categorize it local features and global features. It is not trivial to find the optimal feature combination for a specific task. Though, the classical algorithm for getting the handcrafted feature is still inspired researchers to make improvement on point clouds deep leaning. Thus, getting some of the most representative point cloud feature algorithms will help to understand the further development of the current state-of-art point cloud neural network.

3.1.1 Point Feature Histogram

The PFH(Point Feature Histogram)[13] extend the previous work of a statistical 3D-shape representation for rapid classification to provide the local 3D point cloud features. The implementation of this algorithm is Iterate over points in the point cloud P, which the point in the point cloud are all have per-vertex normal. For every point P_i (i is the iteration index) in the input cloud all neighbouring points within a sphere around Pi with the radius r are collected. This set is called P_ik (k for k neighbours). Then, the loop regards pairs of two points in P_ik, say $p1$ and $p2$. The point whose normal has the smaller angle to the vector $p1 - p2$ is the source point ps, the other one is the target point pt. 3.2

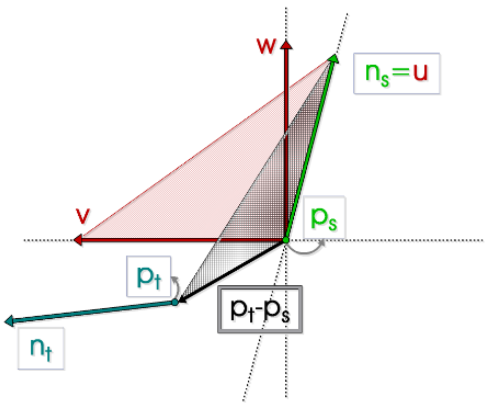


Figure 3.1: Illustrate the Point Feature Histogram algorithm.[7]

3.1.2 Fast Point Feature Histogram

A more powerful version of PFH, so called FPFH(Fast Point Feature Histogram)[12], It works a lot alike with the previous work on PFH, But there are some optimisation steps making FPFH faster. The first approach is that the loop only regards pairs of point P_i with each of its neighbours (remember in PFH the loop would generate pairs of P_i and its neighbours and also between Pi_s neighbours. furthermore the distance between P_s and P_t was left out from FPFH, which is maintained in FPH. As in FPFH only direct pairs between the query point Pi and its neighbours are taken into consideration (far less computation) the resulting histogram is called SPFH (Simple Point Feature Histogram). To compensate the "lost" connections the neighbours' SPFHs are added to Pi's SPFH according to their spatial distance, this method is newly added into the FPFH.

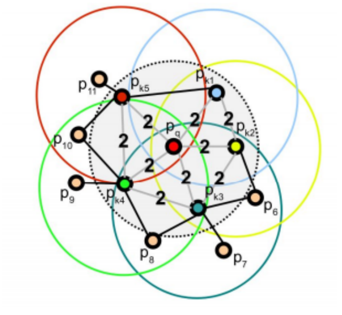


Figure 3.2: Illustrate the Fast Point Feature Histogram algorithm.[7]

3.2 3D CNN network

Most work on 3D CNN networks convert 3D point clouds to 2D images or 3D volumetric grids.[18][9] proposed to project 3D point clouds or shapes into several 2D images, and then apply 2D convolutional networks for classification. Although these approaches have achieved dominating performances on shape classification and retrieval tasks, it is nontrivial to extend them to high-resolution scene segmentation tasks. [23, 9, 6] represent another type of approach that voxelizes point clouds into volumetric grids by quantization and then apply 3D convolution networks. This type of approach is constrained by its 3D volumetric resolution and the computational cost of 3D convolutions.[11] improves the resolution significantly by using a set of unbalanced octrees where each leaf node stores a pooled feature representation. Kd-networks[4] computes the representations in a feed-forward bottom-up fashion on a Kd-tree with certain size. In a Kd-network, the input number of points in the point cloud needs to be the same during training and testing, which does not hold for many tasks.

3.3 Pointnet:achieve effective feature learning directly on point clouds

The prerequisite of 3D CNN networks are input data prepossessing, which mean that 3D CNN have to have the 3D point cloud is converted to other representations before its fed to a deep neural network. In that case people are wondering that is there any way we can achieve the feature learning directly on a different input representation for 3D geometry using simply point clouds.

In the paper PointNets[8], they provide a new idea of effective feature learning method that directly apply the raw data without processing of point clouds. Point clouds are simple and unified structures and are easier than meshes to learn from. The PointNet, however, still has to respect the fact that a point cloud is just a set of points and therefore invariant to permutations of its members.

From an aspect of point cloud's data structure, a point cloud is an unordered set of vectors. While most works in deep learning focus on regular input representations like sequences (in speech and language processing), images and volumes (video or 3D data). One research work from Oriol Vinyals et al[21] that mentioned in PointNet's paper looks into this problem. They use a read-process-write network with attention mechanism to consume unordered input sets and show that their network has the ability to sort numbers. with the inspiration from previous work, the PointNet research team design a deep learning framework that directly consumes unordered point sets as inputs. A point cloud is represented as a set of 3D points $\{P_i \mid \text{i} = 1, ..., \text{n}\}$, where each point P_i is a vector of its (x, y, z) coordinate. Additional dimensions may be added by computing normals and other local or global features. Here, the key is the use of a single symmetric functiona. Symmetric function takes n vectors as input and outputs a new vector that is invariant to the input order. For example, + and operators are symmetric binary functions. As shown in experiments 3.1, they find that applying a MLP directly on the sorted point set performs poorly, though slightly better than directly processing an unsorted input. Effectively the network learns a set of optimization functions/criteria that select interesting or informative points of the point cloud and encode the reason for their selection. The final fully connected layers of the network aggregate these learnt optimal values into the global descriptor for the entire shape as mentioned above (shape classification) or are used to predict per point labels (shape segmentation).

	Accuracy
MLP(unsorted input)	24.2
MLP(sorted input)	45.0
LSTM	78.5
Attention sum	83.0
Average pooling	83.8
Max pooling	87.1

Table 3.1: Three approaches to achieve order invariance. Multilayer perceptron (MLP) applied on points consists of 5 hidden layers with neuron sizes 64,64,64,128,1024, all points share a single copy of MLP. The MLP close to the output consists of two layers with sizes 512,256

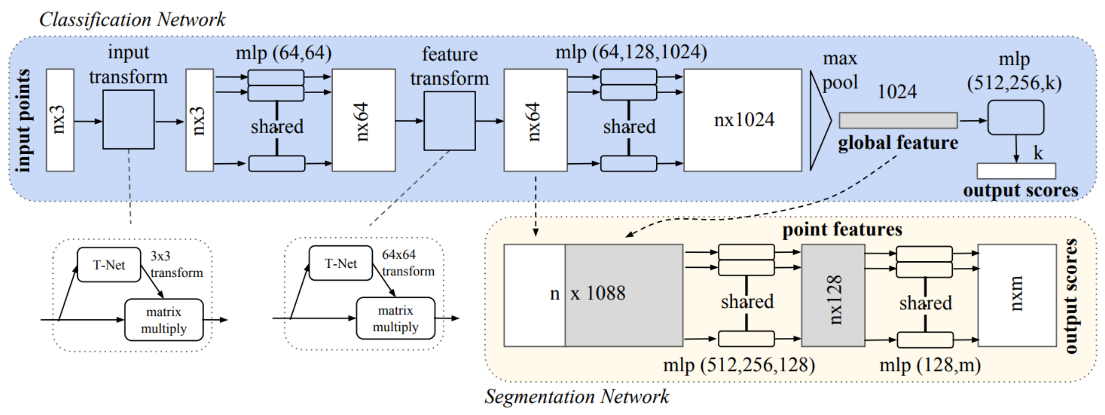


Figure 3.3: PointNet Architecture[8]. The classification network work on categorizing single or multiple objects in one point cloud scan, we will not discuss the classification task in this thesis, but to give a general introduction to the semantic segmentation. The segmentation network is an extension to the classification net, the goal of semantic segmentation is to categorize each point in the scene into m different classes. It takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling, then it concatenates global and local features and outputs per point scores. mlp stands for multi-layer perceptron, numbers in bracket are layer sizes. Batch norm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net. The output is per point scores for m categories.

People can easily train a SVM or multi-layer perceptron classifier on the shape global features for classification. However, point segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner. Their solution can be seen in 3.3 (Segmentation Network). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

3.4 PointNet++:further enhance local feature in matrix space

Although PointNet is a pioneer of this direction with surprisingly well performance, however, by design PointNet does not capture local structures induced by the metric space points live in, limiting its ability to recognize fine-grained patterns and generalizability to complex scenes. PointNet++, they introduced a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. By exploiting metric space distances, this new network architecture is able to learn local features with increasing contextual scales.

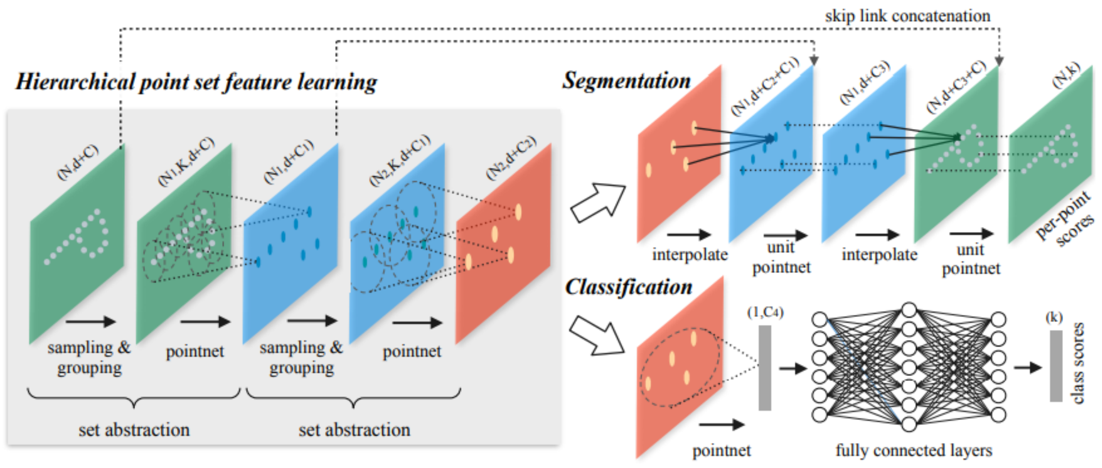


Figure 3.4: PointNet++ Architecture[10]. Illustration of our hierarchical feature learning architecture and its application for set segmentation and classification using points in 2D Euclidean space as an example. Single scale point grouping is visualized here. For details on density adaptive grouping

While PointNet uses a single max pooling operation to aggregate the whole point set, the new architecture builds a hierarchical grouping of points and progressively abstract larger and larger local regions along the hierarchy. The new structure consists of a number of set abstraction levels 3.4. At each level, the set abstraction level is made of three key layers: Sampling layer, Grouping layer, and PointNet layer.

A set abstraction level takes an $N(d+C)$ matrix as input that is from N points with $d-dim$ coordinates and $C-dim$ point feature. It outputs an $N_0(d+C_0)$ matrix of $N^{'}$ subsampled points with d-dim coordinates and new $C^{'}$ -dim feature vectors summarizing local context.

The Sampling layer selects a set of points from input points, which defines the centroids of local regions. Grouping layer then constructs local region sets by finding neighboring points around the centroids. PointNet layer uses a mini-PointNet to encode local region patterns into feature vectors.

Chapter 4: Workflow and System Design

BIM as a technology-based process has made it easier for the involved experts to collaborate using a virtual 3d Model of the building. BIM is centered around value information and many deliverables are produced from model information such as:

Model-based renderings, Energy analysis, Clash and flaw detection, Digital handover, Commissioning documents.

All of these data need to be formed well because it can put a strain on internal systems and procedures if it is not managed properly. In advance to its construction to manage data and linking to the 3D Model, a data-based platform, including all the graphical and nongraphical information regarding the building process. This platform can be an online place for collecting, managing and sharing information amongst a team working on a project. can avoid the potential complications that may arise from the use of individual models and objects that are produced by different project teams.

One of the focus of our system is on the development of tools that can provide to help users through the process of visualizing model and getting quality data management. Manage and transfer data flows in our system is essential for data collaboration and vital to BIM success.

4.1 System Design

To understand the landscape and needs of a specific task. Our work allows us to provides an efficient data management, visualization solution, agile communication system, and user-friendly interface.

Figure4.1 illustrate the major nodes for achieving the goal with their individual function,

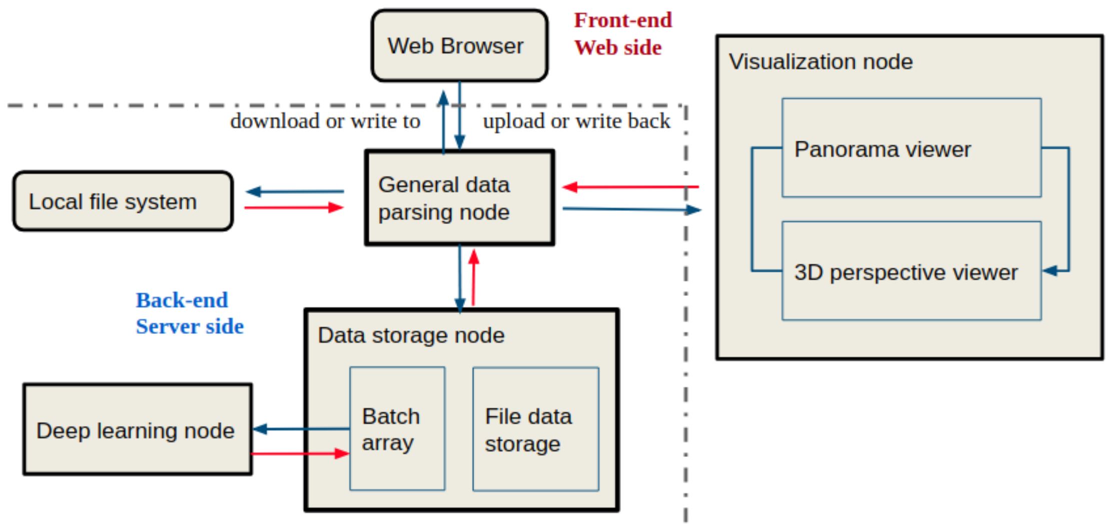


Figure 4.1: This graph illustrate the major functionality module and how data go through all these module in our system.

The whole system can be divided into a front-end function part and the back-end part. The front-end side mostly works on providing user interaction interface and showing point cloud representative visualization in three different approaches. The front-end interaction interface provides several operations that sending to back-end layer, this operations includes storing and parsing data format, the way of determining how to parsing the data will automatically be made by the "General Data Parsing Node". Figure4.1 shows that the file can be uploaded into the system by uploading online or directly maintained by the administrator in the server's file system, after other operations get the result, the user can download the result or data that have been processed(such as intermediate file that parsed from different format, semantic segmentation result or 2D panoramic image). General data parsing node is one of the core function modules in our system, it enables to use of multiple different formats of the input file, which have compatibility to extend the range of file format.

The data flow transmission throughout the whole system by using an intermediate file, it makes sure that for each function module, the requirement of input format is not the main hindrance and allows all the modules to use the most appropriate file format in order to lower the programming difficulties and the compatible language problem. The intermediate file communication method makes sure the system focuses on the portability and reusability and achieves the low coupling which is convenient for farther development and reusing of the code.

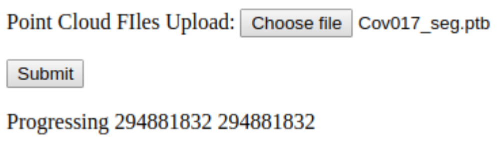
After the data been parsed by "General Data Parsing Node", the formed data will store in the data storage node. The data in general parsing node has been parsed into two structure, the first structure in 2D array panorama form, this form list the point cloud information in 2D array structure as same as 2D panorama image sequence. The second structure is in batch array format, then we can directly put the batch array into neural network, in order to train the network for further improvement or just calculate the semantic segmentation result with one scan input point cloud data by going through the forward propagation. For the detail of how data the formed in all these intermediate files, we will discuss in the next chapter.

For the highlight in this system is our "Deep Learning Node", the Deep learning node is an abstract interface that can trigger the deep learning related algorithms and neural networks get in a run. The choice of the neural network's model can be changed, either replace or update, it shows the extension ability of the system. Researchers can replace or upgrade the model when they want to have a different model architecture of network in order to make the performance further improvement, or have a new model that can working on a different task.

Another highlight feature in this system is "Visualization Node", which provides the method to visualize the raw data and the analyzed result that applied to the raw data views. Panorama viewer and 3D perspective viewer can provide two different way for users to visualize the point cloud data and task result. As for Panorama viewer, it is now providing the ability to show the raw data panorama image with original color, and the instance segmentation result processed by "Novana" algorithm. It provides several useful tools that can assist in organizing the point cloud data and making annotation operation on the panorama image directly through the canvas. The 3D perspective viewer provides the primary visualization window to demonstrate the point cloud model in perspective view.

4.2 Showcase and functionality introduction

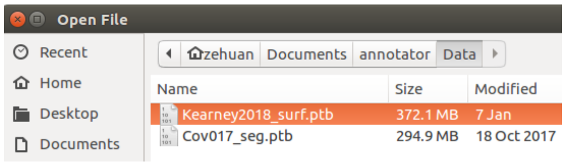
The operation with user friendly interaction will lead user to go through the whole pipeline that finish their task step by step.


Point Cloud FIles Upload: Choose file Cov017_seg.ptb

Submit

Progressing 294881832 294881832

Figure 4.2: Process of using file uploader



Name	Size	Modified
Kearney2018_surf.ptb	372.1 MB	7 Jan
Cov017_seg.ptb	294.9 MB	18 Oct 2017

Figure 4.3: Process of using file select

In 4.7, 4.8, these two figures explain the very first step that user upload a data file to the server, then internally the file parsing node on server side will reform the data.

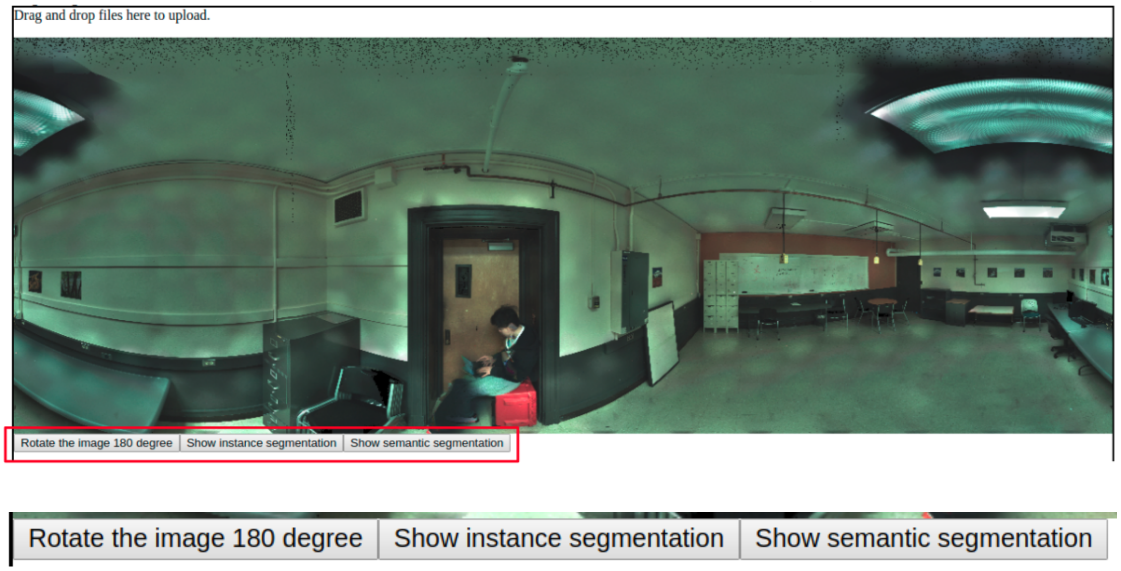
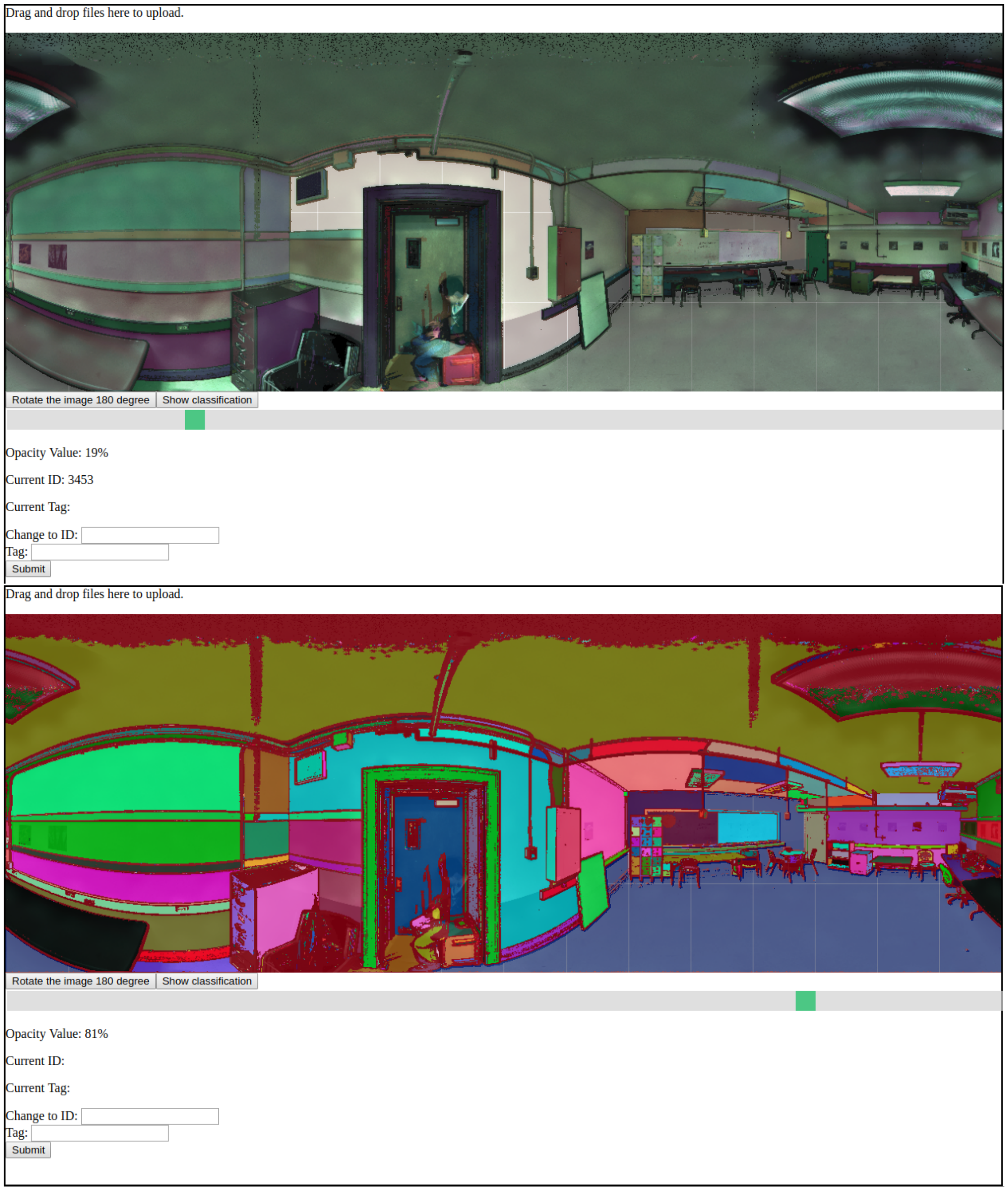

Drag and drop files here to upload.
Rotate the image 180 degree
Show instance segmentation
Show semantic segmentation
Rotate the image 180 degree
Show instance segmentation
Show semantic segmentation

Figure 4.4: The primary stage after the first step, this figure shows the raw color image of panoramic.

After data has been reformed, the panorama image viewer will pop up and show a panorama image with original color and with three buttons to allowing following operations, which contains the image rotation operations and two other buttons to trigger the functions ”show instance segmentation” and ”show semantic segmentation”.



Drag and drop files here to upload.

Opacity Value: 19%

Current ID: 3453

Current Tag:

Change to ID:

Tag:

Drag and drop files here to upload.

Opacity Value: 81%

Current ID:

Current Tag:

Change to ID:

Tag:

Figure 4.5: Instance segmentation result show on the canvas overlay of the original color image.

In figure 5.5, the result of instance segmentation will show up and generate the canvas overlay on the original color image. The instance segmentation result generated according to algorithm[2], test building data collected by Erzhuo in ”Covell Hall”, Oregon State University. This panoramic photograph in different image fuse ratio (opacity). The interface also includes the panel to show which color block is indicate which class ID, in that case, people can easily access the classification result directly from the algorithm and do manual annotation or fix back the result, this change we made could be used to further improving the algorithm or annotate the dataset with real ground truth classification result and feed into deep neural network.

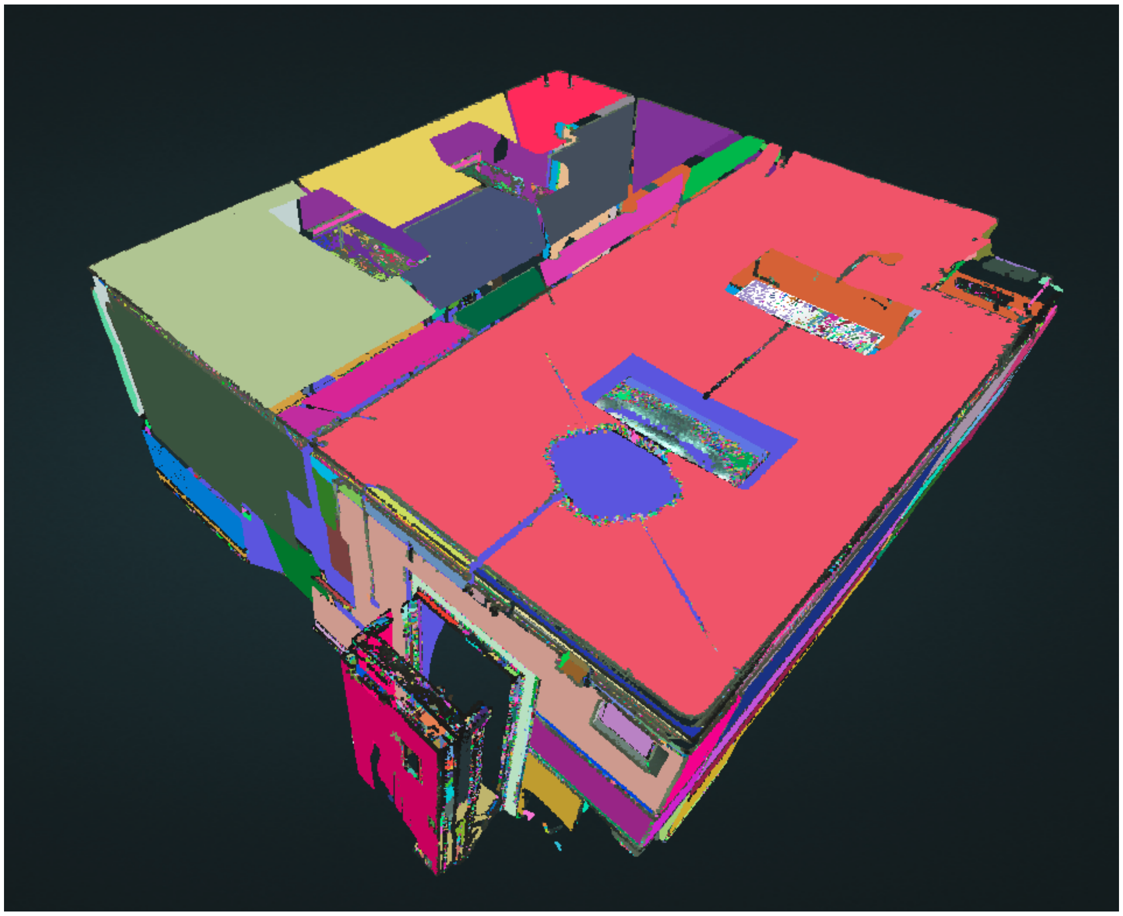
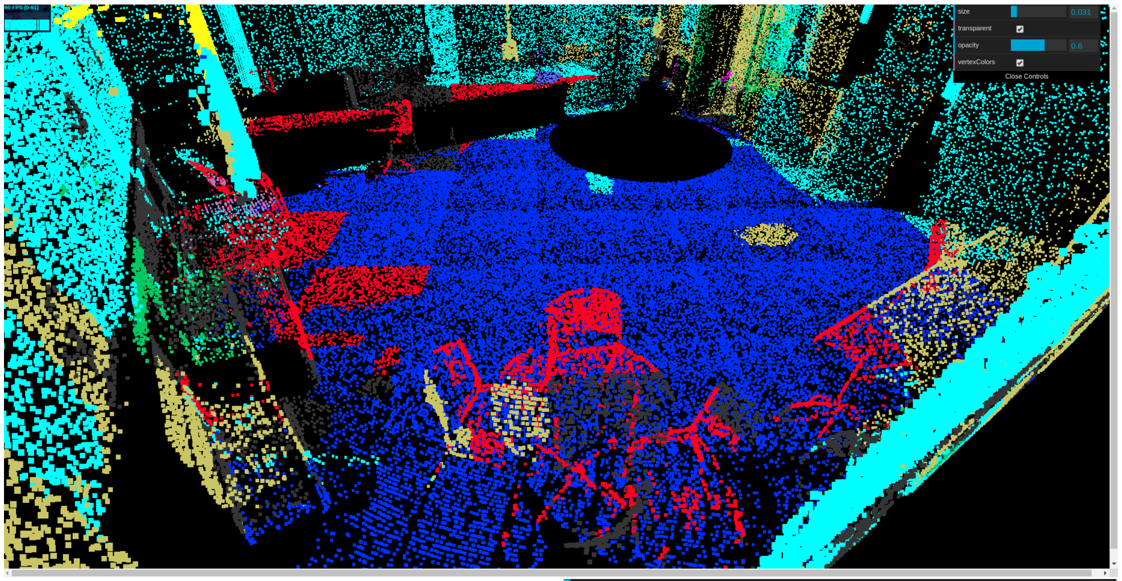
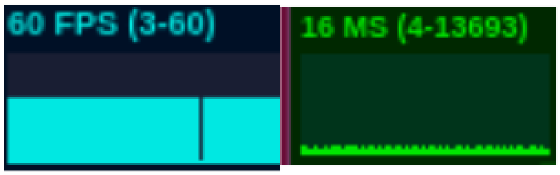


Figure 4.6: The 3D perspective viewer shows the instance segmentation result

Along with the appearance of 2D panorama with instance segmentation overlay and tools, the 3D viewer of instance segmentation result is also shown up. Orbit navigation with mouse and keyboard are supported, this viewer powered by Nested octree data structure, points are rendering in hierarchy order such that the rendering per frame will dramatically be decreased otherwise it will drain down the browser's resources and leading to system crash.

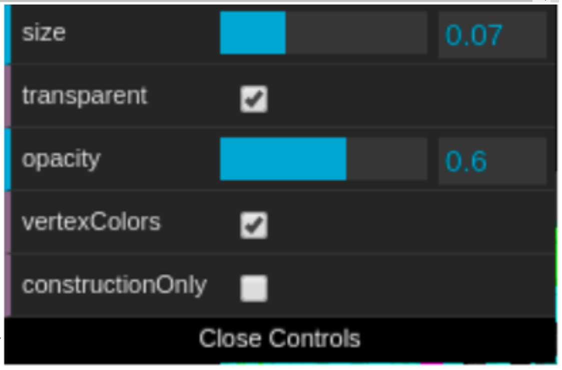




60 FPS (3-60)

16 MS (4-13693)

Figure 4.7: Status monitor window, to show the refreshing rate and calculation ability per second.



size 0.07

transparent ✓

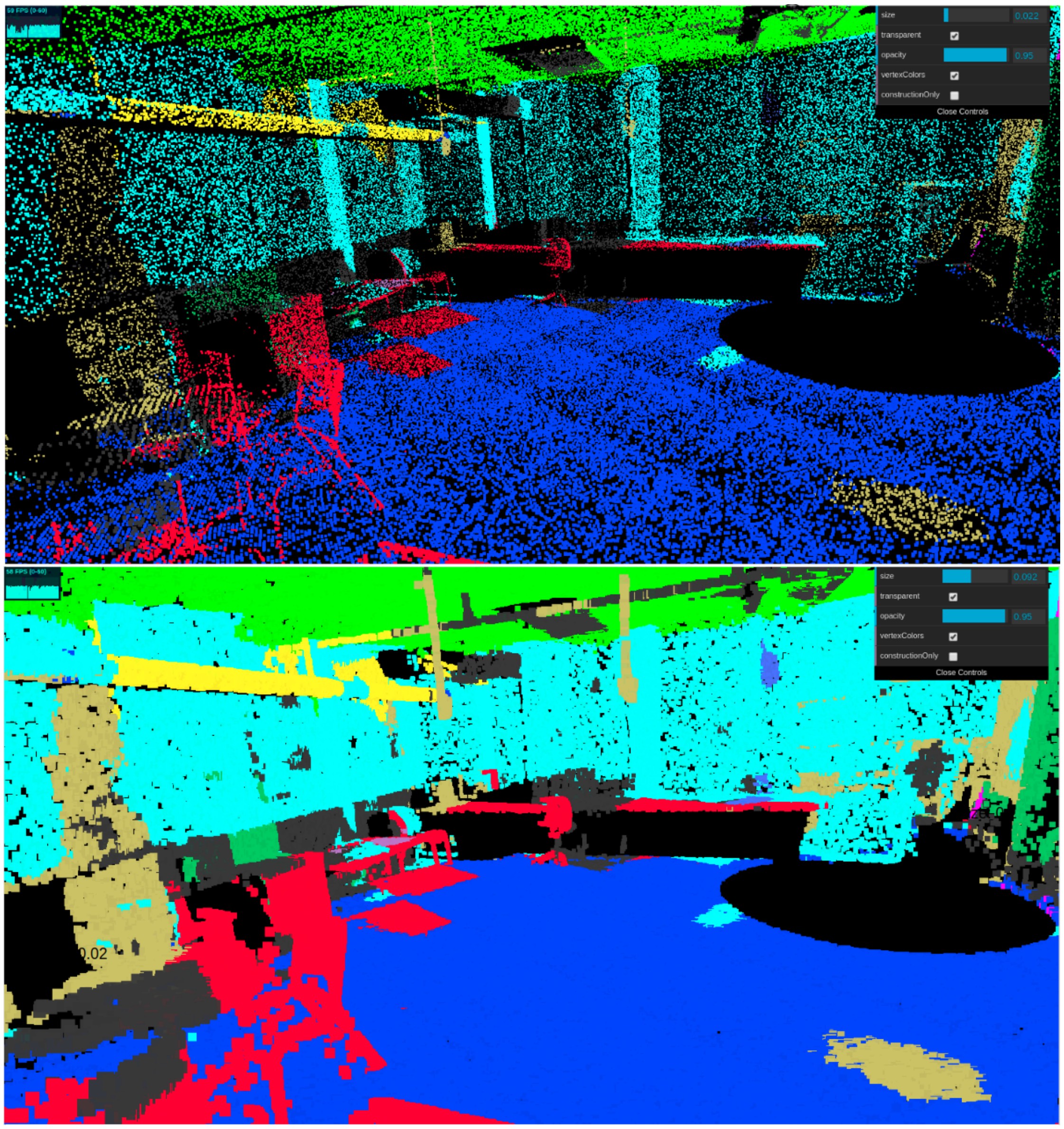
opacity 0.6

vertexColors ✓

constructionOnly

Figure 4.8: Control panel to make visualization status change, including opacity, point size, color shader and construction points filter.

After the user clicks on the semantic segmentation button, the result of scene semantic segmentation will show up through the 3D perspective view. The interface also includes the control panel to adjust the point size and opacity. On the left side of the window, the status element is monitoring the frame per second and a total number of points parameters.


size 0.022
transparent ✓
opacity 0.95
vertexColors ✓
constructionOnly
Close Controls

size 0.092
transparent ✓
opacity 0.95
vertexColors ✓
constructionOnly
Close Controls

Figure 4.9: Point size comparison, between 0.02(figure on top) to 0.1(figure on bottom). To adjust point size for point cloud, sliding the bar in control panel.

The point cloud is much sparser than the point cloud shown in 4.6 due to the data structure to load the point cloud is no longer octree, and the result of point cloud with the semantic result is down-sampled by the algorithm.

Chapter 5: Implementation and Features

5.1 WebGL and Three.js

WebGL[3] is OpenGL branch applied on web browsers, which is intended to provide high-performance computational 3D rendering capabilities to web pages on a wide variety of devices, including desktop, laptop, and mobile devices. Till OpenGL ES 2.0, to ensure that WebGL applications will run on as many devices as possible.

Some features in Our system, such as point interpolation, orbit navigation, make use of Three.js[20] rendering library to handle scene graphs and draw calls. Three.js is a cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser. Three.js uses WebGL. Direct use of the WebGL API and GLSL are limited to special cases where three.js does not provide some necessary functionality, but the agile programming style and capability on the browser makes it inevitable support in our work. Due to the advent of WebGL, Three.js allows the creation of GPU to accelerated 3D animations using the JavaScript as part of a website without relying on proprietary browser plugins.

5.2 Asynchronous and Parallel Execution

Javascript code, no matter what kind of library package, is executed in a single thread. Javascript applications provide callbacks function that is triggered by the browser when certain events, the events such as mouse click, timeouts, or user input, have been met. The web browser will handle the updates and render on the page, in between the execution of callbacks.

Javascript and other languages can offer asynchronous functions to monitor the event. The callbacks to be invoked at a specific time, or to proceed specific event in parallel. The addEventListener function, for example, monitor callback after an event activity has been detected.

The asynchronous version of the XMLHttpRequest is used heavily in Ajax programming, this function tells the browser to load a resource and invoke a callback during progress or after loading has finished. This enables a Web page to update parts of content on a page without interrupting the program is running. The source loading itself is done in parallel. Multiple invocations of HTTP requests may result in multiple resources being loaded simultaneously.

The Fetch API provides an interface for fetching resources (including across the network). Similar to XMLHttpRequest, but the new API provides a more powerful and flexible feature set.

A browser decides when to invoke the rendering callback based on several conditions. For rendering and visualizing the point clouds 3D view result, the callback will usually try to maintain 60 frames per second or less if the current web browser tab is active, otherwise, when the tab is hidden or running in the background, the browser may not invoke the callback in order to reduce CPU usage.

The asynchronous callback philosophy is no also applied to the main loop while file loading is processing. Loading and transmit the data file is a time-consuming operation, due to the size of point sets data are huge and the transmission speed is highly correlated to the local network transmission status. In our work, the file loader function will not block other operation such as rendering, annotating or segmentation. The size of the raw data file corresponds to the number of points that those files contain, we will go through the details of data storage in the next section.

5.3 Data Storage

Our system consists of front-end web user interface module, back-end data storage, back-end python micro framework server communication module and deep learning forward computation module. The combination of all different function module makes this Web-based system an End-to-end style. The user only required for a minimum understanding of the technique, such as image processing, network communications and deep learning training.

In order to achieve the end-to-end architecture and ensure file format variety, the combination of inter-module communication with general file transfer function module is a good option. A metadata file specifies which attributes are available. The attributes are the Euclidean coordinates of a point, color attributes RGB, intensity and classification.

The format is based on the file format of the data stream, with the main difference that floating-point coordinates are quantified to, and stored as fixed-point values.

The two main kinds of input data are stream data and structured data. The stream data, typically, has a less compact data structure that not easy for us to extract the data directly through the data attributes, this storing method is mostly using bit encoding to compress the real data, the information of how to form the data is recorded in the header that represents in first several lines of the file. Take .ptb file as an example, shown in table 5.1.

Attribute	Format	Bytes
Total number of pixels	uint32	4
Header:row	uint32	4
Header:column	uint32	4
XYZ	float	3*4
RGB	uchar	3*4
intensity	float	2
classification	uint32	4

Table 5.1: data stream form in ptb file

As for structured file, either using library compatible file parsing function that form the data to a highly structuring file object in programming language, for example, the file object of H5py in Python, or form the data to structuring by lines and columns in file, such as the .obj and .txt format for instance, in table 5.2.

header	attributes
v	X , Y , Z , R , G , B
vn	X , Y , Z
m	Material file name in string

Table 5.2: data form in obj file, in each line

5.4 Panoramic image viewer

In some specific scanning technique, the raw data of scanning will have a different format to maintain the point cloud data in an original way. Point clouds are great at representing the current state of an area. However, sometimes they have difficulty to work with. It is not uncommon that in such big data set like point cloud its hard to identify all objects correctly. For various reasons, the shape of the object might appear distorted because of missing points on its surface. Sharp edges are especially hard to represent and the quality of that representation depends very much on scan resolution. Fortunately, these inconveniences can be reduced. Its worth to capture a panoramic image along scanning the area. The panoramic image can add meaningful context to collected data and fill in the information lacking in the point cloud, this can save you a lot of time and effort when trying to recognize objects from the point cloud.

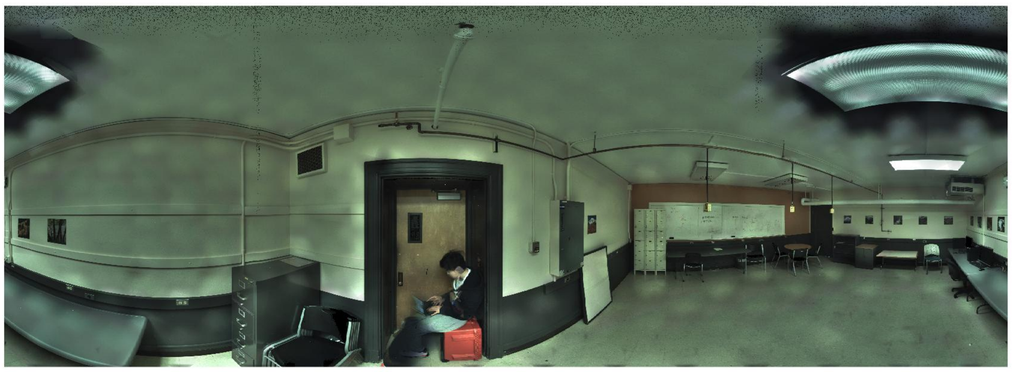


Figure 5.1: Panoramic photograph filmed from Covell building, by Erzhuo.

5.4.1 Color map conversion

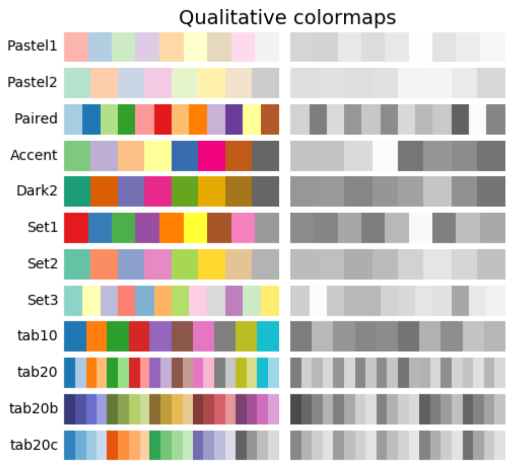
For our applications, in order to visualize the panoramic image classification result, a perceptually uniform colormap is the best choice, one in which equal steps in data are perceived as equal steps in the color space. The idea behind choosing a good colormap is to find a good representation in 3D color space for your data set. The best colormap for visualizing data set classification depends on many things including the total number of classes, colormap uniform distribution.

In our panoramic viewer, we want to demonstrate classification result on the image in a different color overlay, due to the total number of classes in each input dataset can be different, it is tough to promise the colormap for each instance of data to distribute uniformly by picking up color from existed colormap.

Conversion from grayscale is an option to generate a colormap for each specific dataset, which is having a different amount of classes in total. Some of the better ones use a linear combination of the RGB values of a pixel, but weighted according to how we perceive color intensity. A nonlinear method of conversion from grayscale is to use the lightness L^*. in general, similar principles apply for this question as they do for presenting ones information perceptually; that is, if a colormap is chosen that is monotonically increasing in L^* values, it will represent reasonably to grayscale colormap, and then we can map the grayscale in each RGB channel to form a 3D colorspace for our data set.

Algorithm 1 GENERATE COLORMAP FROM GIVEN TOTAL NUMBER OF DIFFERENT CLASSES, MAPPING TO RGB CHANNEL

```
n=total number of classes
$C^*$ = [ ] represents colormap
for i in range of n do
    k,r,g,b = i,0,0,0
    for j in range 8 do
        r | = (k & 1) shift to left 7-j bits
        g | = ((k & 2) shift to right 1 bit) shift to left 7-j bits
        b | = ((k & 4) shift to right 2 bits) shift to left 7-j bits
        k shift to right 3 bit
        append [r,g,b] to C
    end for
end for
return $C^*$
```



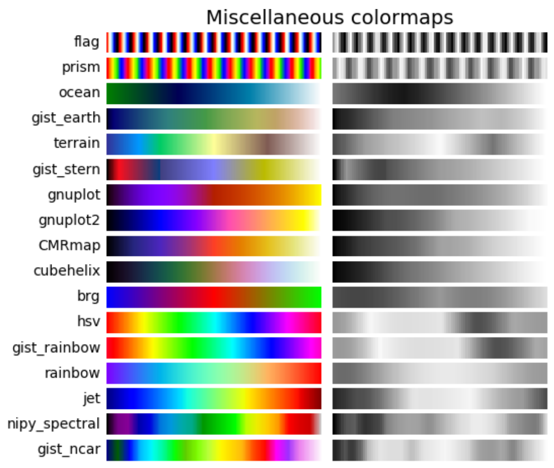


Figure 5.2: Qualitative colormaps[5]

Figure 5.3: Miscellaneous colormap[5]

Figure 5.4: Examples of two main colormap cotegories and the convertion between grayscale to three-dimensional color space

With the colormap has been generated, the classification blocks can be assigned to the colormap according to the colormap's index pair with class index. Furthermore, to avoid two neighboring visualization block have closed color (depending on the result of different classification algorithms, the two neighboring blocks may have their classification index consecutive.), shuffle the colormap's [r,g,b] list can be an optional operation. The result of an example of classification visualization in the same room as 5.1 shown in the following. 5.5

5.4.2 Tools for interaction

The panoramic image viewer can not only visualize the result of the classification result, but it can also allow users to interact with a mouse click directly or using tools on the control panel. In order to make the demonstration of the classification result applied to the raw input data, the viewer also allows the controller to manipulate with opacity classification result overlay, thus of original RGB panoramic image layer can be shown during the demonstration process and makes user have deeper understanding of how the classification algorithm works on the original dataset.

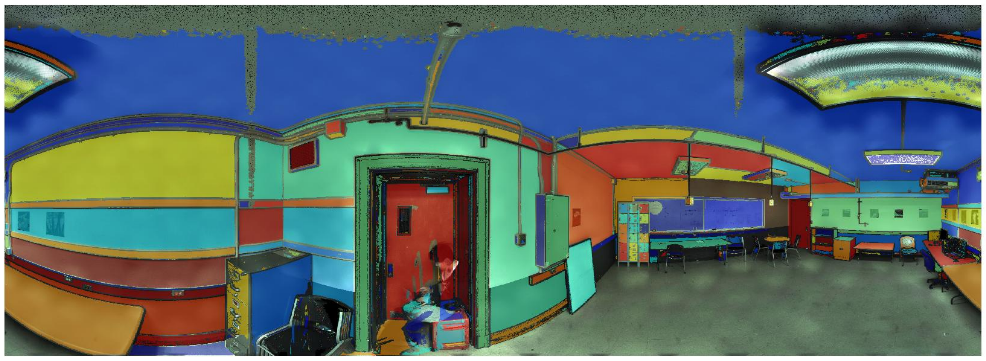


Figure 5.5: The result of classification algorithm of Covl building panoramic photograph.

In controller tool panel, the sliding bar is taking charge of changing the fusing ratio between the original color image panorama and classification overlay panorama, range from 0%(only original color) to 100% (fully display the classification color map)

This interaction panorama viewer powered by Javascript canvas elements, we use a multi-layer canvas structure to build-up an element overlapping structure. This structure allows us to do the visualization and interaction at the same time. The reason why using multi-layer canvas architecture instead of only use single layer to do both visualization and interaction is that any update applies on the canvas require the total redraw on it, the one-click operation, for example, will spend time on doing multiple time of repeating draw on the same canvas. The panorama data display and mouse hover operation indicator will have to play in a sequence, which is a waste of computation resources and it will cause the system running very slow when data set increases.

5.5 3D geometry viewer and related tools

5.5.1 Point size adjustment

Determining the point size is an important factor for speed and visual quality. A low point size improves the performance and reduces occlusion between points, but it also leads to holes in the rendered images. A larger point size reduces holes between points, but it also reduces performance and increases occlusion artifacts.

In our system, it provides the user interface element dat.GUI, in order to maintain the 3D material of each point. The material's properties include point size and opacity of points. We also applied point cloud filter button to trigger the visualization of scene semantic segmentation result, in this specific scenario, the building structure in the scene can be displayed individual without showing the furniture and clutters.

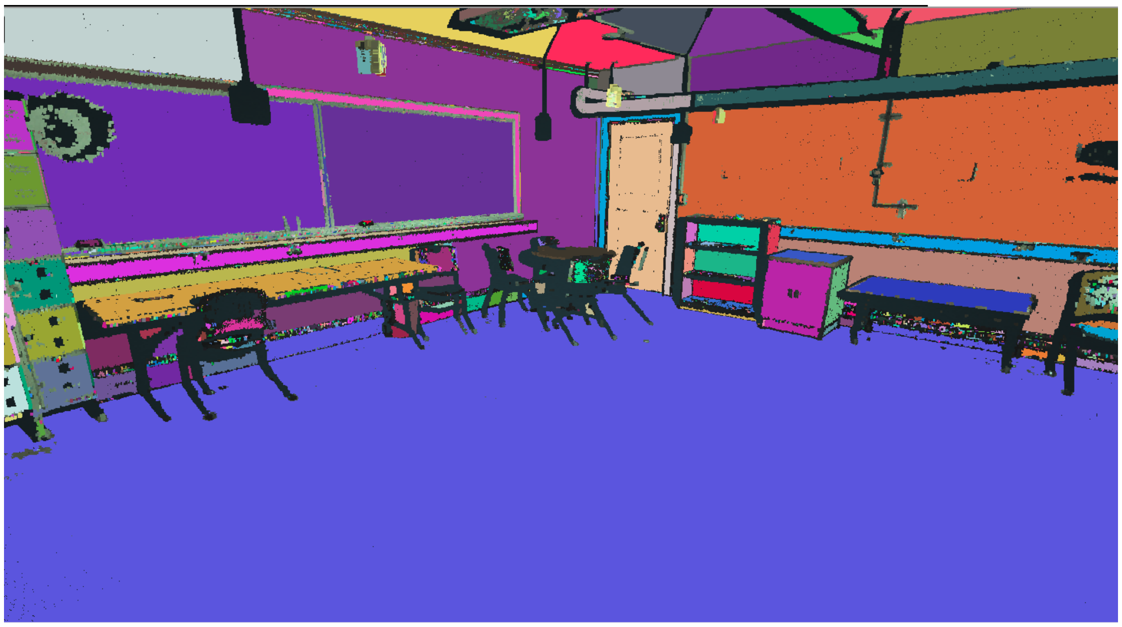


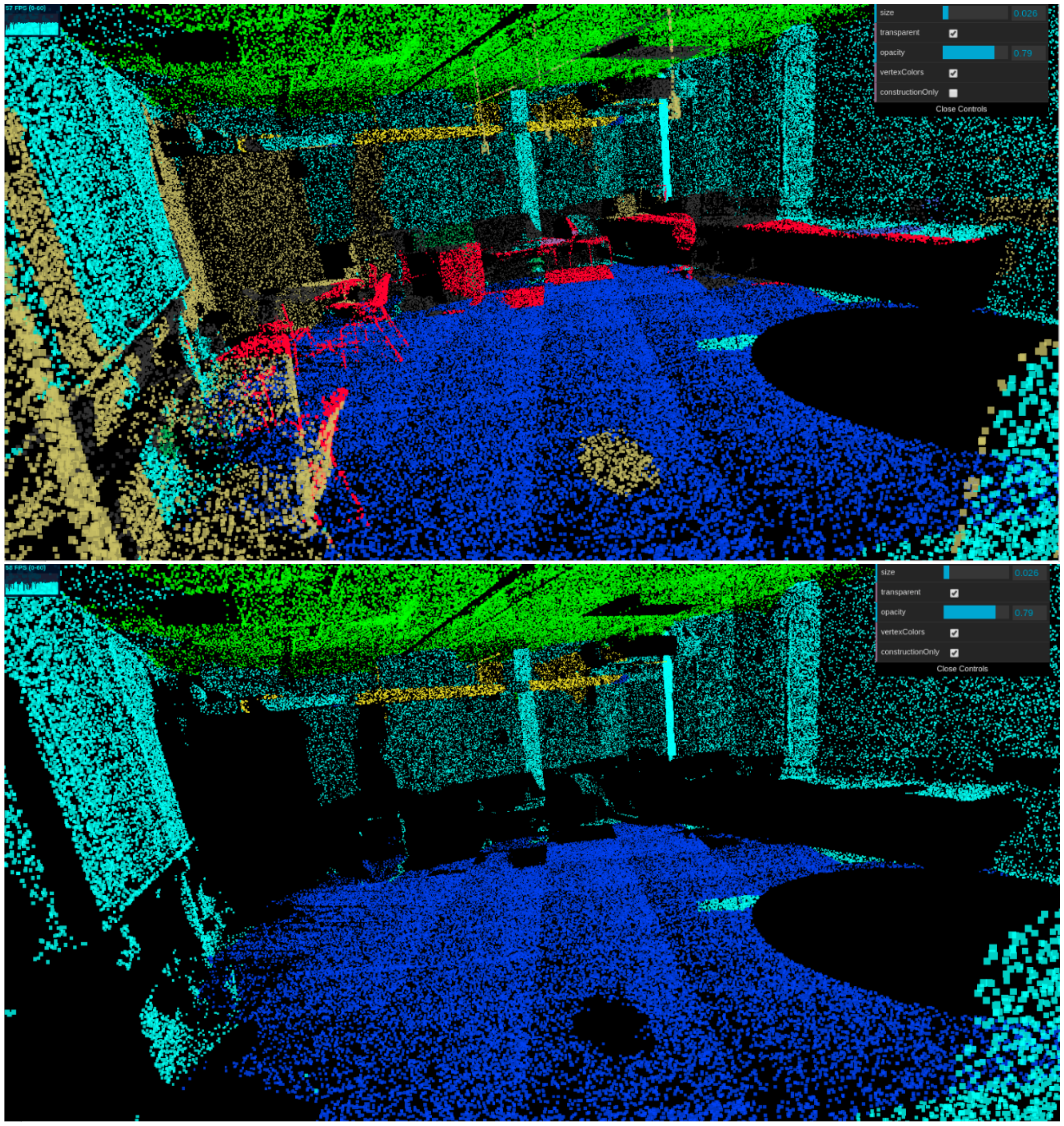
Figure 5.6: This figure shows the result of instance segmentation algorithm of Covl building in 3D perspective view, point rendering color is depending the classification result, it shows the different color in each classified instance.

5.5.2 Semantic scene element filtering

With the observation from point set, the semantic segmentation result will approximately sample the points into 13 classes, which listed in the below form. We can further distinguish these 13 classes as construction object(contain such as floor, ceiling and wall etc) or non-construction object(contain such as table, sofa and door, etc). With the tag of a different class, we can achieve that the visualization of two clusters, which is construction and non-construction, and all different object one of each or check the list to show in various combinations.

Construction classes	Non-construction classes
ceiling	sofa
floor	chair
wall	table
beam	door
column	window
	bookcase
	board
	clutter

Table 5.3: 13 semantic classes used in our approach.


size 0.026
transparent ✓
opacity 0.79
vertexColors ✓
constructionOnly ☐

size 0.026
transparent ✓
opacity 0.79
vertexColors ✓
constructionOnly ✓

Figure 5.7: This figure shows the result semantic segmentation result with construction structure filter function, with one click on control panel, the 3D viewer will give visualization of point cloud that distinguished by their class.

Chapter 6: Experiments

6.1 Performance

This chapter mainly discusses the detail of our experiments and show some of the samples of the result. The visualization on the 3D viewer shows the dataset from S3DIS(3D Semantic Parsing of Large-Scale Indoor Spaces)[1].

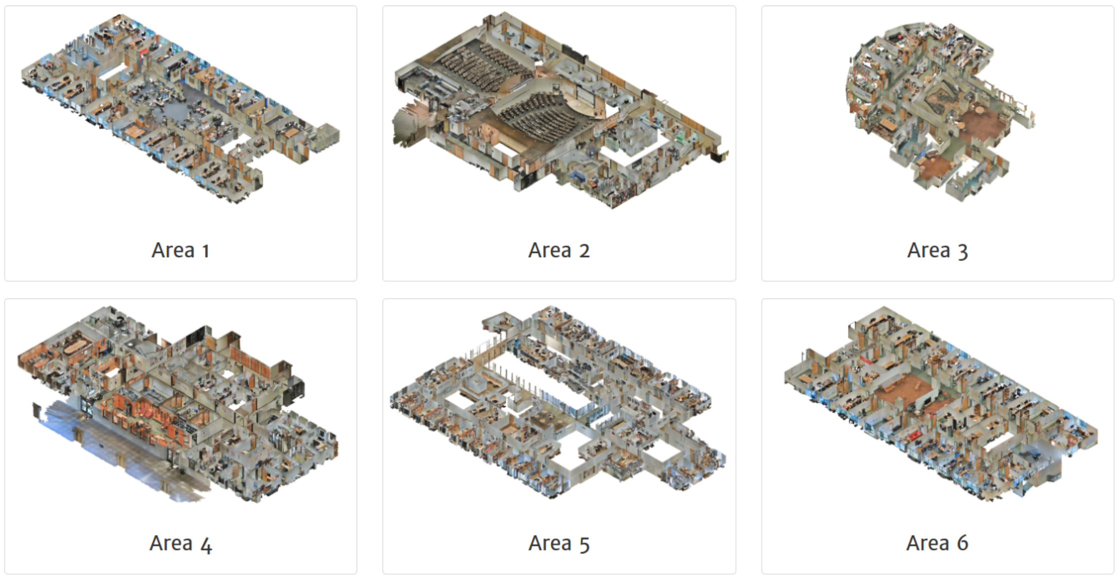


Figure 6.1: This figure shows the sample of S3DIS dataset point cloud collection area[1]

The S3DIS dataset is composed of five large-scale indoor areas from three different buildings, each covering approximately 1900, 450, 1700, 870 and 1100 square meters respectively (total of 6020 square meters). These areas show diverse properties in architectural style and appearance and include mainly office areas, educational and exhibition spaces, and conference rooms, personal offices, restrooms, open spaces, lobbies, stairways, and hallways are commonly found therein. One of the areas includes multiple floors, whereas the rest have one. The entire point clouds are automatically generated without any manual intervention using the Matterport scanner.

Their dataset detects 12 semantic elements, which are structural elements (ceiling, floor, wall, beam, column, window and door) and commonly found items and furniture (table, chair, sofa, bookcase and board). Notice that these classes are more fine-grained and challenging than many of the semantic indoor segmentation datasets. The dataset contains 3D scans from Matterport scanners in 6 areas including 271 rooms. Each point in the scan is annotated with one of the semantic labels from 13 categories (chair, table, floor, wall etc. plus clutter).

The training data points are were split by room, and then sample rooms into blocks with area 1m by 1m. We train our segmentation version of PointNet to predict per point class in each block. Each point is represented by a 9-dim vector of XYZ, RGB and normalized location as to the room (from 0 to 1). At training time, we randomly sample 4096 points in each block on-the-fly. At test time, we test on all the points. We follow the protocol of 6-fold strategy for train and test.

In figures 6.2,6.3,6.4,6.5,6.6 demonstrate five different test set indoor scenes, selected from S3DIS dataset. Each figure contains ground truth semantic labeling and the point cloud predicted label of different classes.

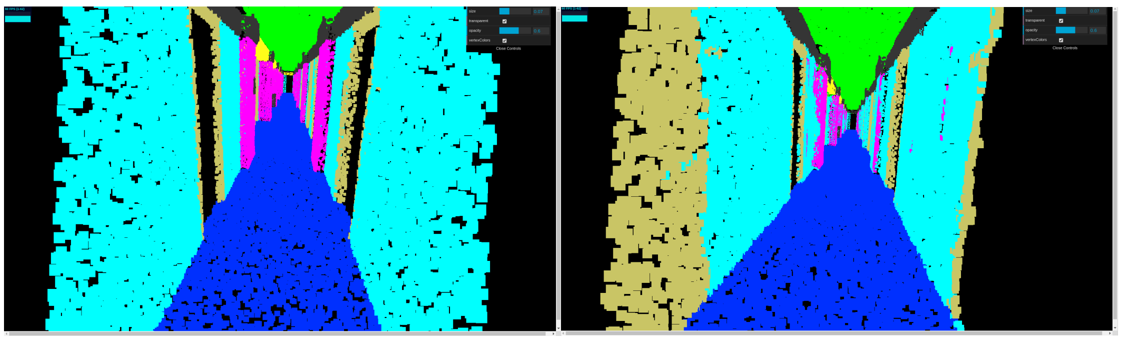


Figure 6.2: Hallway point cloud in S3DIS, the left one is ground truth scene and the right image shows the prediction result by using our approach.

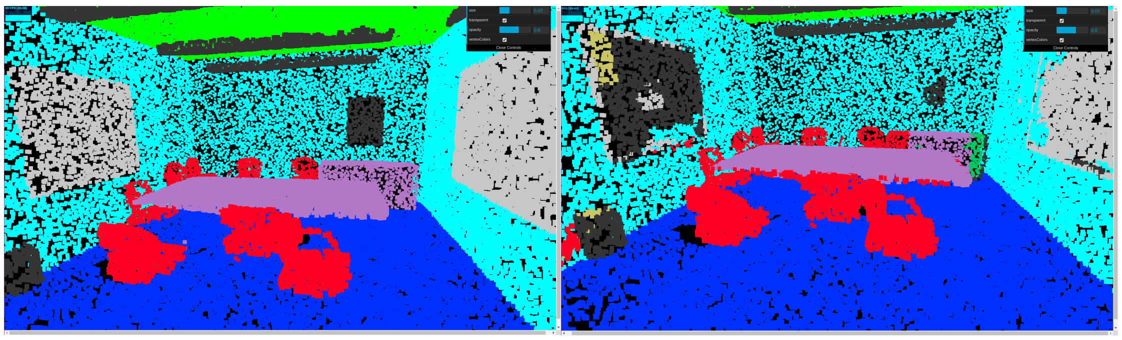


Figure 6.3: The conference room result comparison, the left image shows the ground truth labeling and the right image shows the prediction result by using our approach.

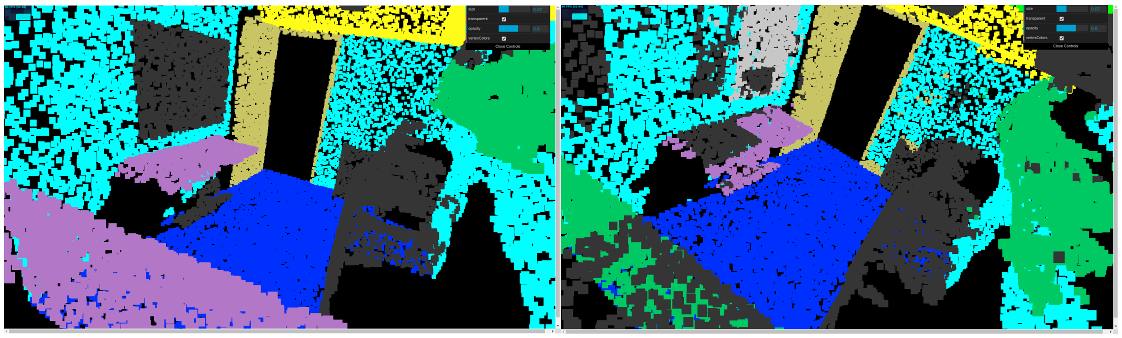


Figure 6.4: The copy room result comparison, the left image shows the ground truth labeling and the right image shows the prediction result by using our approach.

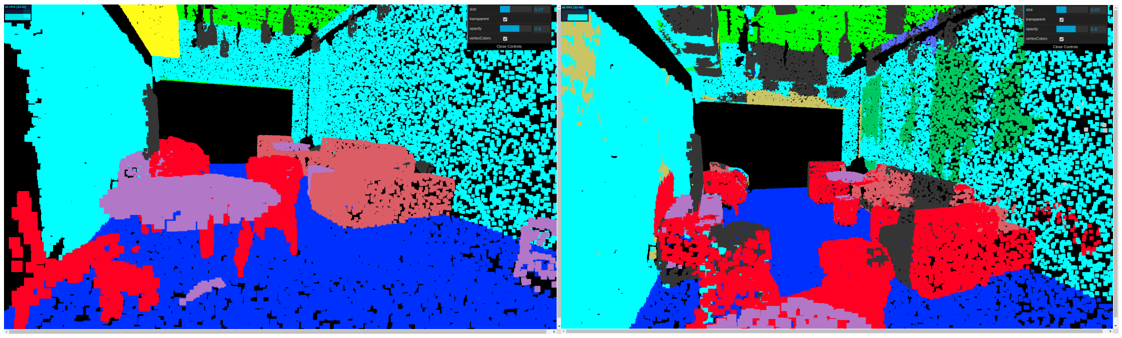


Figure 6.5: The left image shows the ground truth labeling of lounge data scene and the right image shows the prediction result by using our approach.

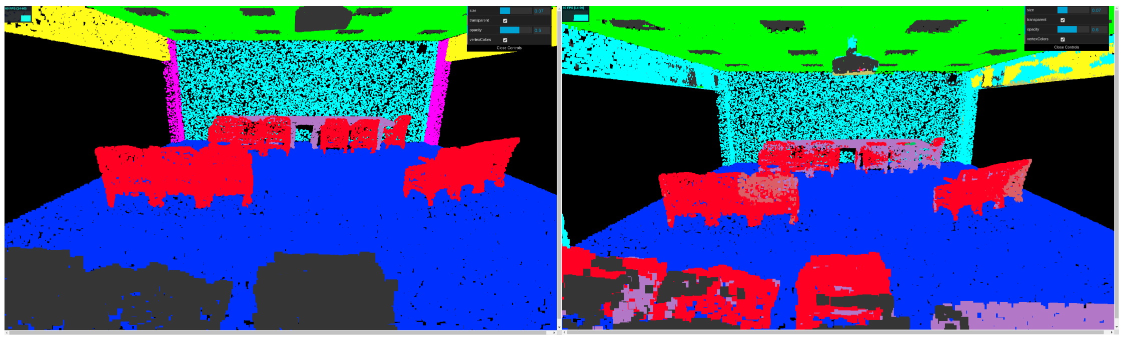


Figure 6.6: An open space in S3DIS, the ground truth labeling image shows on the left and the prediction result is shows on the right.

For the indoor scene like Hallway in figure 6.2, it has a well formed construction layout, without having much corner and details of small object the segmentation of walls, floor and ceiling are have very good result, which the figure also demonstrate that the prediction result is highly close to the ground truth. In Figure 6.3, the board and clutter can easily be mismatched, the white board on the wall and the paint on the wall (belong to clutter class) are not well performed in this result, but for tables and chairs are distinguished by PointNet with a little mismatch. In the image pair of Figure 6.4, most of the labelling are well done in copy room scene, but we can still found some flaws, such as the table are mismatched with the part of ceiling on the button-left corner of the figure. The false classification is also appear in 6.5 both chair and sofa. The basic of labeling is well formed, except some points on the wall will be treated as ceiling. The chair and sofa also have mismatched result, we can see the point of sofa show the color of class chair. In figure 6.6, when column and beam are close to the wall may cause a very difficult segmentation, especially when the color are looks very similar(RGB channel are less useful in this scenario).

6.2 Construction structure evaluation result

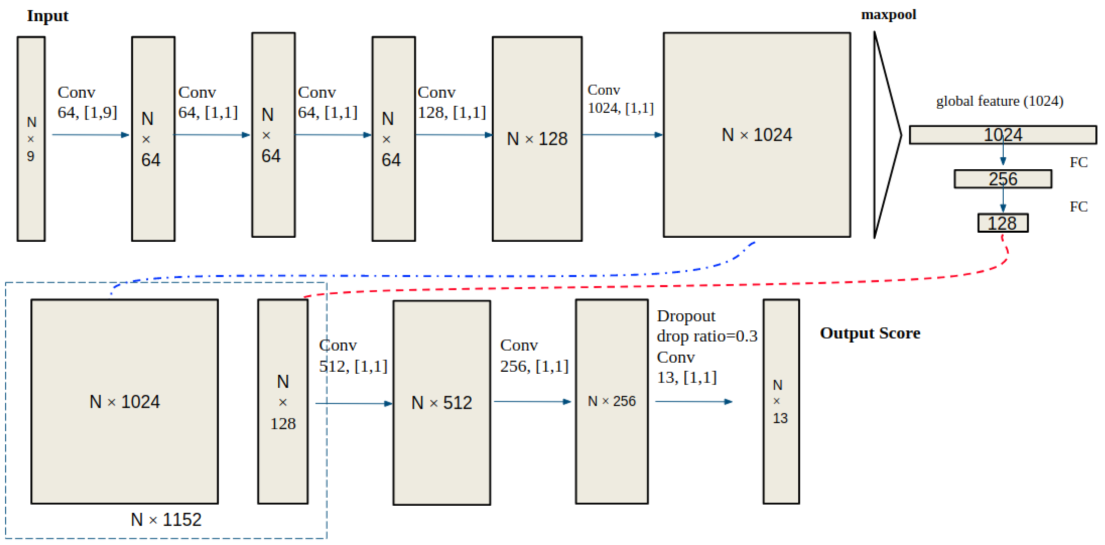


Figure 6.7: The practical neural network model architecture demonstration.

In this work, PointNet[8] architecture is utilized in the deep learning function node 6.7. The structured batch array, which contains the information per point in indoor point cloud scene, will be fed to the proposed network. Each point has 9 attributes, including X, Y, Z, R, G, B, normalized location with respect to the room (from 0 to 1).

The architecture have 3 consecutive multiple layer perception (MLP), in our implementation, we use convolutional filter with size of 11 instead. After the feature learning and get the 1024 new point features, we applied a max pool layer to down-sample and aggregate these point features into global feature, then make the global feature go through fully connected layer (FC) to aggregate to 128 features. With 128 features and tile it to the same dimension as number of points, then concatenate it with the previous N1024 features. After the next several convolutional layer that similar to previous operations, we get the output score of 13-dimensional array after softmax layer.

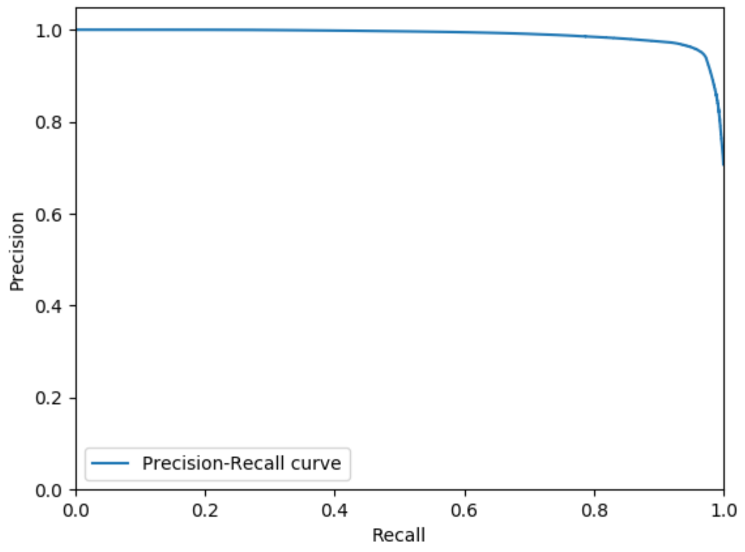


Figure 6.8: Precision-Recall curve for structural and non-structural objects classification in the S3DIS dataset

The precision-recall curve of construction structure point and non-construction structure point, this result is transferred from initial 13-class semantic segmentation for each point into binary classification.

Construction classes	Non-construction classes
ceiling	sofa
floor	chair
wall	table
beam	door
column	window
	bookcase
	board
	clutter

Table 6.1: 13 semantic classes used in our approach.

Chapter 7: Conclusion and future work

We presented deep segmentation user interactions that are able to render point clouds with millions of points in real time in standard web browsers. The intermediate data storage format and data stream parsing method ensured that the system has high compatibility, which means it can process regardless of the input file format file and neural network architecture. Besides, the utilizing of asynchronous mechanism and recall function in front-end dramatically reduce the total waiting time and improve the runtime speed, which is running the main loop in parallel. The general file storage node and multiple views approach enable our system to render points cloud raw data in several different representatives, including panorama and 3D viewer, thereby increasing the way of making deeper interactions with point clouds data set. To achieve a better visual quality, we implemented methods such as multi-layer canvas interaction interface, to increase the computing efficiency and enable visualization and input interaction process in parallel. We have also shown a deep semantic segmentation result on indoor scene point cloud dataset S3DIS. Distinguish the point set as construction and non-construction label for each point in visualization. In the future work, we will keep refining the system efficiency in point cloud rendering and add up more interaction tools, including the synchronous correspondence of two viewers. Furthermore, we can also make point cloud selection tools in the 3D perspective viewer. Filling the "holes" after we filter out the non-construction object in building scan is another track we can pursue. More efficient point cloud semantic segmentation algorithms and network architecture will be applied on, the PointConv[22] architecture introducing a state of art performance in doing the semantic segmentation task.

Bibliography

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[2] Erzhuo Che and Michael J Olsen. Fast edge detection and segmentation of terrestrial laser scans through normal variation analysis. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2017.

[3] khronos. webgl. `https://www.khronos.org/registry/webgl/specs/1.0/`, 2015. [Online; accessed 22-Oct-2018].

[4] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 863–872. IEEE, 2017.

[5] matplotlib.org. Choosing Colormaps. `https://matplotlib.org/users/colormaps.html#grayscale-conversion`, 2015. [Online; accessed 28-Dec-2018].

[6] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[7] pcl.org. Overview and Comparison of Features. `https://github.com/PointCloudLibrary/pcl/wiki/Overview-and-Comparison-of-Features`, 2015. [Online; accessed 28-Oct-2018].

[8] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[9] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[10] 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*, pages 5099–5108, 2017.

[11] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 3, 2017.

[12] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. Citeseer, 2009.

[13] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 643–650. IEEE, 2008.

[14] M Schutz. Potreeconverter-uniform partitioning of point cloud data into an octree, 2014.

[15] Markus Schütz. Potree: Rendering large point clouds in web browsers. *Technische Universität Wien, Wiedeń*, 2016.

[16] ShareLiDAR.com. ShareLIDAR. `https://www.sharelidar.com/`, 2015. [Online; accessed 22-Oct-2018].

[17] sketchfab.com. sketchfab. `https://sketchfab.com/`, 2015. [Online; accessed 22-Oct-2018].

[18] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[19] Mirage Technologies. PointCloudViz Server. `http://server.pointcloudviz.com/`, 2015. [Online; accessed 22-Oct-2018].

[20] threejs.org. threejs. `https://threejs.org/`, 2015. [Online; accessed 22-Oct-2018].

[21] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[22] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. *arXiv preprint arXiv:1811.07246*, 2018.

[23] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.