ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

SEMANTIC INFORMATION DERIVATION FROM 3D POINT CLOUD

M.Sc. THESIS

Murat KENDİR

Informatics Institute
Geographic Information Technologies Programme

JULY 2019

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

SEMANTIC INFORMATION DERIVATION FROM 3D POINT CLOUD

M.Sc. THESIS

Murat KENDIR
(706131030)

Informatics Institute

Geographic Information Technologies Programme

Thesis Advisor: Asst. Prof. Dr. Caner GUNEY

JULY 2019

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

3B NOKTA BULUTU İLE SEMANTİK BİLGİ TÜRETME

YÜKSEK LİSANS TEZİ

Murat KENDİR
(706131030)

Bilişim Enstitüsü

Coğrafi Bilgi Teknolojileri Programı

Tez Danışmanı: Dr. Öğr. Üyesi Caner GÜNEY

TEMMUZ 2019

Murat KENDIR, a M.Sc. student of ITU Graduate School of Informatics Institute student ID 706131030, successfully defended the thesis/dissertation entitled Semantic Information Derivation from 3D Point Cloud”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Caner GUNEY** Istanbul Technical University	....
Jury Members :	**Prof. Dr. Rahmi Nurhan ÇELİK** Istanbul Technical University	....
	Asst. Prof. Dr. Melis Uzar DİNLEMEK Yildiz Technical University	....

Date of Submission : 17 May 2019
Date of Defense : 11 June 2019

To a deaf carpenter, blind tailor and urban planner of highlands...

FOREWORD

20 years ago, when I was working at my father’s woodshop, I was always enjoying with watching my father’s designs and measurement drawings. Accuracy of measurements, designs and real room geometries was always hiding difficult challenges for my father. So, the reality was always a secret enemy of design, but my father was knew it. That’s why measurements was always taken much more time than other wood works. He was always obsessed about measurements and had causes to keep this behaviour, because he was knew that wrong or missing measurements costs always much more time and resources than other faults.
Now, I am 35 years old and keep working as geomatic engineer in different work places such as dam construction, mining facility construction, measurement technologies market and spatial database management. By the way, I always tried to upskill myself at different areas. In 2006, I tried to create an online sector map, which visualizes regularly classified commercial and industrial places in Turkey. In 2008, I started a web project (istanbulkazanben180.com) shows panoramic views at along of Bhosphorus Anatolian and European sides. In 2012, I developed a web-based GIS tool called mymapbase.com, which was first tool visualizes geometries stored in MySQL Spatial Extension. In 2016, I developed a tool to stream real-time cm level accurate location information to web browsers.
In both of professional and avocation times in my life, I couldn’t escaped from 3 dimensional reality and never couldn’t kept myself at safe area of conventional mapping methods.
The other side of medallion showed me that there was nothing special about me, it was just the thing called Zeitgeist. Mapping/geographic information technologies was rapidly spreading on surveying and mapping. There is only one thing special about me; I never forgot my father’s rule “Reality is secret enemy of design”.
Thanks to my family for their patience, my wife Esra for compete with me and my advisors Caner Güney, Rahmi Nurhan Çelik for their supports.

June 2019 Murat KENDIR
(Geomatic Engineer)

TABLE OF CONTENTS

ABBREVIATIONS

ALCF(D)	:Attributive Language with Complements (Descriptive)
ASCII	:American Standard Code for Information Interchange
BBOX	:Boundary Box
BIM	:Building Information Modeling
BREP	:Boundary Representation
C++ / CPP	:Classes Plus Plus (Programming Language)
CAD	:Computer Aided Design
CM	:Coarse Model
CPU	:Central Processing Unit
CSV	:Comma Separated Values
FME	:Feature Manipulation Engine
GIRAPIM	:Gestión de Información relacionada con la Análisis Previo de las Intervenciones en Monumentos
GIS	:Geographic/Geospatial Information System
GML	:Geographic Markup Language
GPU	:Graphic Processing Unit
GUI	:Graphic User Interface
IFC	:Industry Foundation Classes
IO	:Input – Output
MDL	:Minimum Description Length
LDP	:Location Determination Problem
LSM	:Least Squares Method
GIS	:Geographic Information System
OWL	:Ontology Web Language
OWL-DL	:Ontology Web Language – Description Logic
PCD	:Point Cloud Data
PCL	:Point Cloud Library
RANSAC	:Random Sample Consensus
RDF	:Resource Description Framework

WKB	:Well Known Binary
WKT	:Well Known Text

SYMBOLS

D	:Descriptive
//	:Comments starts with this tag
< >	:Data Type stag in PCL code blocks
::	:Subclass of any other class in PCL code blocks

LIST OF TABLES

Page

LIST OF FIGURES

Page

SEMANTIC INFORMATION DERIVATION FROM 3D POINT CLOUD

SUMMARY

Laser scanner technologies takes off rapidly during last ten years and responses many different industrial solutions, such as documenting as-built constructions. In many situations, real constructions, buildings and heritage areas should be documented with laser scanner technologies and point cloud data should be stored with their real world coordinates and measurements. On the other hand, point cloud data is not easily readable and interpretable by humans. In most scenarios, point cloud data is abstracted by humans with help of Computer Aided Design (CAD) and/or Geospatial Information System (GIS) based software tools. In last several years these processes displaced by semi-automatic object recognition softwares and extensions, such as PointCAB, LASTools, Dielmo3D and etc.

Building Information Modeling (BIM) is another rapidly developing area, which is also a multi-disciplinary application area like GIS. Civil engineers, electrical engineers, surveying engineers, architects and all experts about construction and building management systems are related with BIM concept. Nowadays, in many country BIM based building construction is declared as mandatory for public buildings and/or business centers, because BIM is not only profitable at construction progress, it is also profitable at building management and maintenance processes.

As-built drawings and documentary is also an essential part of BIM. Many contractors in developed countries already use laser scanners to control the production during every implementation phases, such as concrete casting, earth compaction, model-fitting and etc. In undeveloped countries and old buildings at developed countries are also in same situation that are not well documented or does not have floor plans or models. It means there is also a need to document and redraw plans of these buildings. As-built plans of these buildings are created mostly with laser scanners or handheld laser distance measurement devices . Naturally most time consuming part of these

processes is the evaluation process that is usually done manually by human beings human work, which includes drawing interpreting and reclassifying objects.

In the scope of the study, an automatic methodology to recognize walls, floor and ceiling surfaces in a point cloud data has been proposed.

This methodology includes also defining ontological meanings of surfaces, registering to a semantic structure and creating relationships. This progress is only focused on walls, floor and ceiling geometries in a full room without considering other additional indoor objects such as lightings, windows, radiators and other furniture elements.

3B NOKTA BULUTUNDAN SEMANTİK BİLGİ TÜRETME

ÖZET

Lazer tarayıcı teknolojileri son 10 yılda hızla popülerleşti ve çok sayıda endüstriyel soruna çözüm olarak kullanılmaya başlandı. Bunlardan biri de yapıların belgelendirmeleri ve plan çiziminde kullanılan ve “olduğu gibi” anlamına gelen “as·built” çizimlerdir. Bilindiği gibi pek çok farklı durumda yapıların, binaların ve tarihi kalıntıların lazer tarama teknolojileri kullanılarak belgelenmiş olmaları gerekir ve toplanan nokta bulutlarının gerçek koordinatları ve ölçümleri ile birlikte sayısal ortamda saklanması gerekir. Diğer taraftan toplanan nokta bulutu verisi insanlar tarafından kolay okunabilir ve yorumlanabilir nitelikte değildir. Çoğu durumda nokta bulutu verileri CAD / CBS (GIS) yazılımları kullanılarak insanlar tarafından sadeleştirilir, çizime dönüştürülür. Son yıllarda bu sadeleştirme işlemleri yarı-otomatik obje tanıma yazılımları ile yürütülür hale gelmiştir. Bu yazılımlardan birkaçına örnek olarak PointCAB, LASTools, Dielmo3D verilebilir.

Öte yandan, tıpkı CBS gibi, Yapı Bilgi Modellemesi (BIM) de hızla gelişen başka bir çoklu-disiplin uygulama alanıdır. BIM kavramı, inşaat mühendisleri, elektrik mühendisleri, geomatik mühendisleri, mimarlar ve inşaat süreçleri veya bina yönetimi ile ilgili olan tüm diğer uzmanlıkların ortak ilgi alanına girer. Bugünlerde pek çok ülkede BIM tabanlı uygulamalar, kamu binaları ve iş merkezleri gibi çok sayıda insanın içinde bulunacağı binalarda zorunluluk haline getirilmiştir. Bunun birden fazla nedeni vardır. Birincisi, BIM uygulamalarının kullanılmasının, inşaat süreçlerinde eski tarzdaki inşaatlara göre maliyetleri azalttığı bilinmektedir. İkinci olarak da bina inşa edildikten sonra, bina yönetimi ve bakım süreçlerinde görünmeyen masrafları azalttığı görülmüştür.

Uygulanmış çizimler (as-built) ve sayısal belgeler bu perspektiften bakıldığında BIM kavramının önemli bir parçasıdır. Gelişmiş ülkelerde pek çok müteahhit firma beton dökümü, zemin sıkıştırma, model uyumunu test etme gibi inşa aşamalarında lazer tarayıcılarını kullanmaya başlamıştır. Bunun yanında, gelişmemiş ülkelerde binaların

ve diğer ülkelerde de eski binaların belgelendirmeleri yapılmadığı için veya kat planları olmadığı için benzer bir durum söz konusudur. Yani, belgelendirmesi yapılmayan binaların da lazer tarayıcılar kullanılarak uygulanmış çizimlerinin yapılması gereklidir. Uygulanmış çizimler, buna benzer durumlarda lazer mesafe ölçüm cihazları veya mobil lazer tarayıcılar ile yapılır. Doğal olarak bu süreçlerde en çok zaman alan kısım, objelerin tekrar çizimi, yorumlama ve sınıflandırma vb. işleri kapsayan insanların yaptığı çalışmalarıdır.

Bu çalışma kapsamında duvarları, taban ve tavan yüzeylerini nokta bulutu verisi içerisinden otomatik olarak tanıyan bir yöntem önerilmektedir. Bu yöntem, yüzeylerin ontolojik anlamlarını tanımlamayı, semantik veri yapısına aktarılmasını ve objelerin arasındaki ilişkilerin tanımlanmasını içerir. Bu işlemler, dolu bir odada sadece duvar, taban veya tavan geometrilerini dikkate alır; ışıklandırma, pencereler, radyatörler ve diğer mobilyaları göz ardı eder.

Çalışma, PCL (Point Cloud Library) olarak bilinen C++ nokta bulutu işleme kütüphanesi kullanılarak yüzeylerin oluşturulması ile başlamaktadır. Bu kütüphanenin kullanımında çeşitli filtreler kullanılarak nokta yoğunluğu azaltılmış, bu yöntemle yüzeyleri ortaya çıkaran algoritmanın düşük performanslı bir cihazda dahi kolaylıkla çalıştırılabilmesi amaçlanmıştır. Random Sample Consensus (RANSAC) olarak bilinen algoritmanın yardımıyla nokta bulutu içerisindeki en fazla noktaya sahip yüzeyler ortaya çıkartılmış, ardından bu yüzeyler 3 boyutlu çokgen geometrilerine dönüştürülmüştür.

Oluşturulan geometrilerin, duvar, taban veya tavan olup olmadığı oda ile olan ilişkileri test edilerek belirlenmiştir. Filtrelenmiş nokta bulutunun kapsadığı tüm hacim oda nesnesi olarak kabul edilmiş, prizmatik bir katı cisim modeline dönüştürülmüştür. Aynı katı cisim, yüzeylere parçalanarak ve bu yüzeyler etrafında bir tampon bölge belirlenerek, odayı kısıtlayan objelerin, yani taban, tavan ve duvar geometrilerinin bu tampon bölge ile kesişimi değerlendirilmiştir. Bu yolla bu objeler dışındaki diğer tüm muhtemel yüzey objeleri (çalışmada kullanılan veri setinde masa, ışıklandırma ve dolap benzeri nesneler yer almaktadır) filtrelenmiştir. FME (Feature Manipulation Engine) isimli yazılımın yardımıyla, oda orta noktası ile yüzey orta noktaları arasında oluşan vektörün sağ el kartezyen koordinat sisteminde Z eksenindeki bileşeni hangi yüzeylerin duvar, hangilerinin tavan ve hangilerinin taban olduğunu belirlemede kullanılmıştır.

Çalışmanın son aşamasında oluşturulan duvar, taban ve tavan geometrilerinin semantik veri olarak ifadeleri ve sorgulamaları tamamlanmıştır. FME yazılımından oda bileşenleri olarak CSV (Comma Seperated Values) formatında çıkış alınmış, bu formattaki veri Protege isimli ontoloji editörüne otomatik olarak aktarılmıştır. Ardından ontoloji editörü Protege yazılımına eklenti olarak kurulan "reasoning engine"ler (sebep-sonuç ilişkileri kurarak ontolojik sınıflandırmaları yeniden değerlendiren ve bağımsız verileri bu kurallara dayanarak ait oldukları sınıflara atayan yazılım araçları) yardımıyla yüzeylerin ait oldukları ontolojik sınıflara kaydedilmesi sağlanmıştır. Bu yöntemle, semantik veri yeniden değerlendirilmiştir ve üst sınıflar olarak kurgulanan oda, kat ve bina sınıflarına da eş zamanlı olarak kaydolmaları sağlanmıştır. Semantik verinin DL (Descriptive Logic) tabanlı sorgulama yöntemleri (çalışmada SPARQL ve DL Query kullanılmıştır) ile sorgulama örnekleri hazırlanmıştır.

Her ne kadar çalışmada lokal dosya formatları kullanılsa da veritabanı teknolojisinin de kullanılabilirliği değerlendirilmiş, özellikle nokta bulutundan basit geometrik şekillere evrildiği aşama sonrasında, mekansal veritabanı teknolojilerinin de desteklediği ortak bir standart olan WKT (Well Known Text) formatı ile tüm platformlarda kolaylıkla işlenebildiği ve analiz edilebildiği gösterilmiştir.

Süreçlerin tamamında veri girdi ve çıktılarının otomasyona hazır ve herhangi bir veri seti için uygulanabilir olması göz önünde bulundurulmuştur. Bunun için özellikle parametrik ifadelerden, özellikle oda boyutu ile ilgili olabilecek herhangi bir parametre kullanılmamasına gayret gösterilmiştir. Örnek vermek gerekirse, nokta bulutu içerisinden dışa aktarılan, duvar olma ihtimali olan daha küçük nokta bulutları değerlendirilirken, kıyaslandığı diğer nesne yine ilk nokta bulutunun kendisidir. Dolayısı ile yüzeyler herhangi bir büyüklük değerinden ziyade, tarama yapılan odanın içerisindeki konumlarına göre değerlendirilmiştir.

Çalışmanın, son yıllarda gittikçe sayıları artan otomatik semantik veri elde etme çalışmaları ile birlikte değerlendirildiğinde spesifik bir alana odaklandığı söylenebilir, ancak bununla birlikte sürecin tamamının aşama aşama değerlendirilerek ve bir sonraki aşamaya hazır edilerek tasarlanması bakımından ön açıcı olduğu düşünülmektedir.

Farklı sektörlerdeki halihazırda bulunan benzer uygulamaların halen insan yorumu ve müdahalesine ihtiyaç duyduğu göz önünde bulundurarak, gelecekte daha kapsamlı çalışmalar ile birlikte makine ve insan görüşü arasındaki açının kapanacağını düşünülmektedir.

1. INTRODUCTION

Point cloud data is getting more popular by time with current developments on laser scanner technologies. Laser scanner technologies provides ability to collect billions of point coordinates just in several seconds and this number is increasing day by day. Laser scanner technologies improved it’s functionality already in several application areas, such as archaeology, architecture, Building Information Modeling (BIM) and industrial documentation. As a parallel development progress, point cloud data and laser scanner technologies has caused to be created another kind of application area in Geographical Information Systems (GIS): such as automatic object detection, classification and creating topology and relationships between objects. These developments caused to occur of two topics in both of GIS and BIM sector: Scan-to-BIM and Scan-to-GIS. Both topics are very active regarding to developments and technologies provided by both of hardware and software companies.

Creating meaningful and useful information in digital world from massive data collected from reality is a complicated process, just like in regular human life and human interactions with objects. This process basically consists some steps in human brain, such as recognizing objects, classifying it or detect semantic meaning of object and defining relations with other objects. If it is decided to handle all these processes by human manually, it could be caused to spends lots of working hour for humans. Therefore many studies in the world has aimed to develop automatic processes to object detection, object recognition, object classification and creation of relationships among objects.

Researchers from all over the world are focused on object recognition by artificial learning in order to create semantic annotations by using rule-based languages and derive relationships by reasoning engines.

Some studies in the academic literature so far aimed to process point cloud data, reorganize only preferred objects, create semantic structure with a limited vision in the focused area and derive topology and relations between recognized objects. The researchers have accomplished their studies as an end-to-end application. In this study, similar methodology will be applied from larger perspective.

1.1 Purpose of Thesis: Recognizing Walls, Floor, Ceiling and Creating Rooms

Semantic Labeling is most compatible concept between computer vision and human mind. Needless to say, objects can be defined and classified with conventional taxonomy methods, but it is not fitting with reality in every situations. For instance, a room is defined as a closed geometry with walls, floor and ceiling. How about a room with three walls and additionally another wall made with full glass? Is it a room or a terrace? One mono block glass wall can change all ontological meaning and taxonomy of a geometry. In such situations , objects are described with multiple definitions and classified at multiple classes.

Walls, ceilings and floors should be analyzed again with room, flat and building objects after these super classes created. To solve similar problems, it should be defined additionally semantic rules which created by different perspectives such as closeness of shape of rooms, difference between edge number of geometries and consisting walls. On the other hand, uncommon geometries inside buildings can cause to endless different semantic rules to labeling all of them. That’s why it is decided to narrow down the focus to a specific angle with taking risk of ignoring to label uncommon geometries in buildings.

This study concentrates into recognizing walls, floor and ceiling in a room with full of furniture pieces and register these geometries into a semantic structure. The methodology of the study is split in 4 sections as follows, and those sections are defined in detail in the chapter 2:

- Filtering point cloud data
- Extracting surfaces within the filtered point cloud data
- Recognizing walls, floor and ceiling from all extracted surfaces
- Registering them to a semantic structure
- Creating super classes using semantic reasoner

If all these steps can handled by fully automatic processes in computer with minimum parameters, which means there is a possibility to get absolute human readable information from point clouds. As a result of that, this method can be utilized in different application areas that given as below:

- Documentation of old buildings, factories and industrial facilities
- Rapidly modeling with robotic units in dangerous areas or hazardous places (For example, mines, industrial places with hazardous chemical leaks)
- Standardization of as-built drawings in BIM applications for non-documented buildings (especially for buildings with large capacities)
- Documentation and change detection in damaged buildings after earthquakes or other kind of building deformations.

1.2 Literature Review

One of the remarkable researches for similar purposes with this study has been implemented at the Information Management Institute in the Neuchetal University, which is deriving semantic information from point cloud data (Cotofrei et al., 2011). In this work, researchers have aimed to create knowledge derived from point cloud data using Optimal Scene Interpretation method and Minimum Description Length (MSL) principle (Risannen, 1982). Point cloud data was comprised of 320 million points, collected from Pantheon temple and this point cloud was also used in "Karman Center" project. Visual and context information created by following Attributive Language with Complements ALCF(D) concept step by step (Pangercic et al., 2009). RACER and SCENIC applications are able to create these description logic models. One of the distinctive property about this research was the ontology concept, including three parts (see Figure 1.1):

- Coordinate System Ontology: Cartesian, Spherical or cylindrical systems
- Transformation System Ontology: Different transformation methods with different parameter numbers
- Geometric Form Ontology

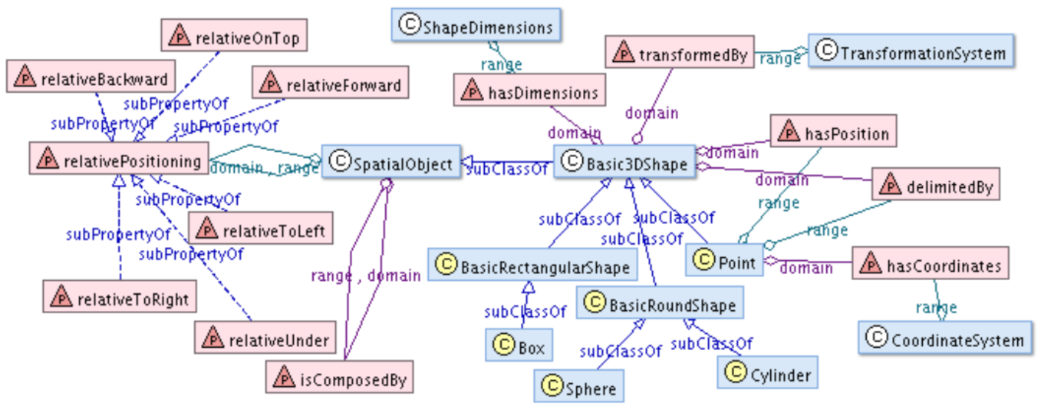


Figure 1.1 : Ontology model used in "Semantic Interpretation of 3D Point Clouds of Historical Objects" project, (Risannen, 1982).

Another work has been developed in Spain to create semantic information from point cloud data in residential place Vallodoid (Finat et al., 2010). Researchers collected point cloud data with Optech Ilris 3D laser scanner and they have also used projected orthophotos to coloring point cloud data. Dominant planes has been detected semi-automatically using UvaCAD software and related extensions (see Figure 1.2). Ontological models created using Resource Description Framework (RDF) format and objects are classified according to Dublin Core standards. Labeling progress has been completed with GIRAPIM library, which developed by DAVAP research group. This library supports also CityGML format. CityGML is a GIS based open source standard which is developed to visualize and manage all spatial objects in urban areas with preserving Level of Detail (LoD) concept, relationship and hierarchy. CityGML also facilitates creating semantic information for all kind of city details (Emgard and Zlatanova, 2007).

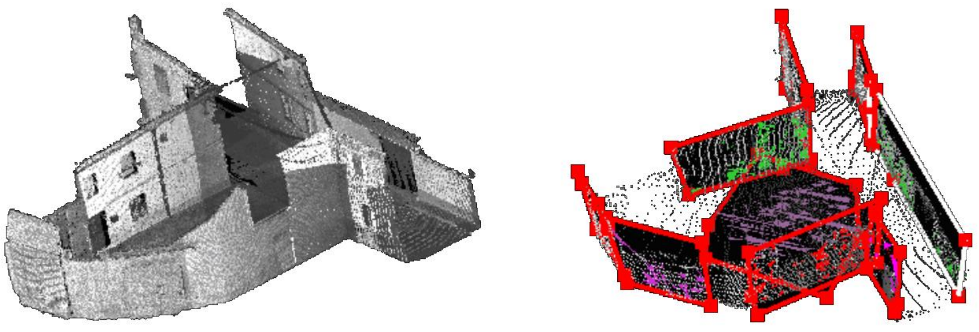


Figure 1.2 : Semi-automatic detection of continuously dominant planes, (Finat et al., 2010).

Another stimulating work aimed to create semantic information for BIM environment in simple architectural objects has been published in the Bourgogne University (Cruz et al., 2007). Main purpose of that project was relating point cloud subsets with Coarse Models (CM) and converting coarse models to IFC standard with help of OWL Plugin of Protege software (see Figure 1.3). It can be also mentioned about an original perspective about this work was two different levels:

- Semantic Level – Descriptions of construction elements in OWL format
- Condition Level – Ontological relationships and topology between construction elements

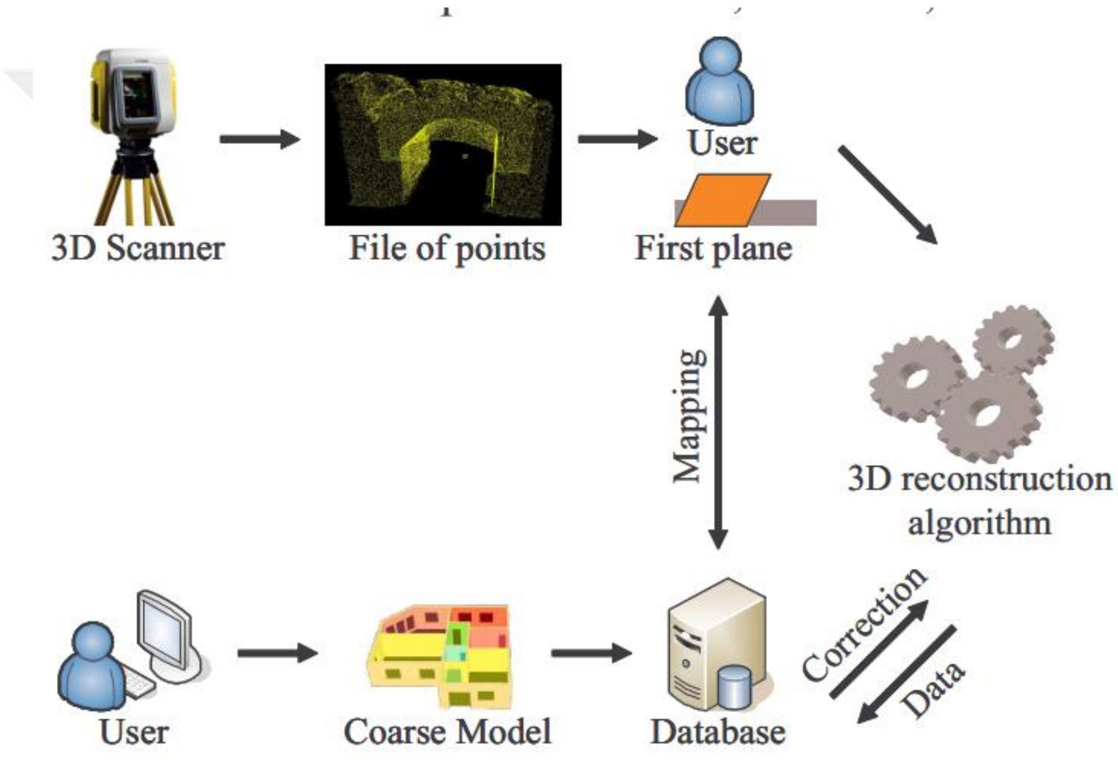


Figure 1.3 : Application schema describes recognizing progress of construction elements using CM, (Cruz et al., 2007).

Recognizing planes and creating relationships with other planes were handled by voxel grid method. Voxel grid sizes are variables defined by user interaction after several tests and voxels are 2D rectangles or 3D cubes consisting points with a minimum size in which algorithms works with an effective speed and main geometry forms are still not deforming. Figure 1.4 represents some of these algorithms, such as detecting differentiation of planes, same planes with gaps, filtering parasitic points and finally creating plane topology according to intersection of voxels.

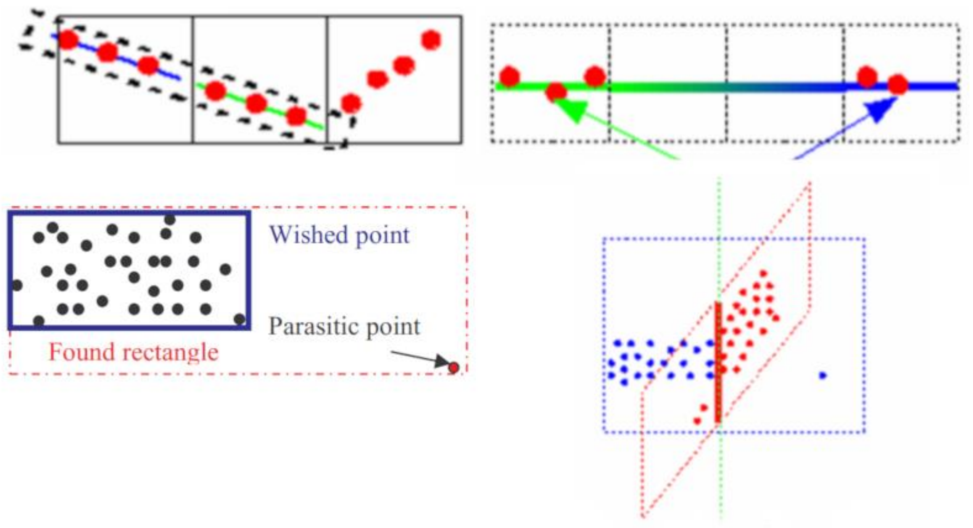


Figure 1.4 : Voxel based filters and plane detection algorithms, (Cruz et al., 2007).

1.3 Hypothesis

Recognizing objects derived from point clouds is popular topic at last decade and related application, library and tool numbers are increasing day by day. These researches and developments also indicate the importance of this need in different areas. On the other hand, semantic technology and in particular ontology driven applications have proved that relationships, topology and hierarchical conditions are also important to recognize and interpret spatial objects. If system architects aim to design a semi-automatic or manual recognition and interpretation progress, the point cloud libraries and applications will be enough to create semantic information. If not, system architects should develop iterative methods to totally automatic create semantic information based on point clouds. This is also similar procedure with human mind before anything recognized, because in most of situation brain is used to compare the relations between objects to recognize them. For example, how people recognize buildings and conclude as semantic meaning of any object is a building? Some semantic rules in human vision are active before they choose semantic meaning of any object like “Buildings are rising mostly above a foundation”, “Buildings have empty spaces”, “Buildings are mostly bigger than average human height”. These rules are indicating that human brain is creating relationships and some topological rules using other related objects (foundation, empty space, human).

In this study, human brain has replaced with computer vision and it is tried to use semantic technology and ontology driven model to recognize preferred objects in the context of a room of a building. Thus, it is suggested to use following steps by order (see Figure 1.5):

- Finding primitive geometries which are wall / floor / ceiling candidates.
- Exporting primitive geometries with additional attributes that are essential to recognizing progress
- Creating semantic classes and super classes by using primitive geometries with their relationships

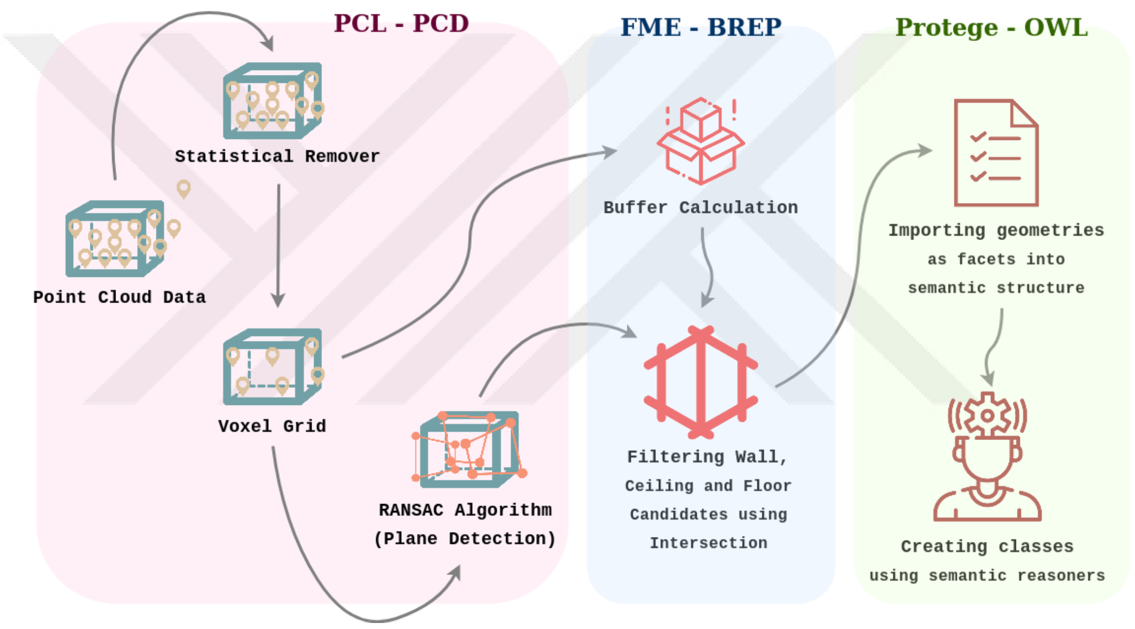

PCL - PCD
FME - BREP
Protege - OWL
Statistical Remover
Point Cloud Data
Voxel Grid
RANSAC Algorithm (Plane Detection)
Buffer Calculation
Filtering Wall, Ceiling and Floor Candidates using Intersection
Importing geometries as facets into semantic structure
Creating classes using semantic reasoners

Figure 1.5 : The schematic view of automatic semantic information derivation progress.

2. METHODOLOGY

To describe methodology briefly and clearly, the semantic labeling structure explained before all the other processes, because the final product from point cloud data should be clearly declared before the other point cloud processes. In the last subsection in methodology, semantic labeling structure will be tested with real values.

2.1 Semantic Labeling Structure

Semantic Labeling created and managed by Protege 5.5.0 with OWL-DL language and Pellet reasoner. Pellet is a Description Logic Reasoner, developed to support OWL language to design a full precision progress automatically for individuals. Pellet is used also for debug and compare ontological information (Gurau and Nüchter, 2013). Web Ontology Language (OWL) is a language designed for representing knowledge and also is able to read by both human and computers. OWL files generally includes classes, hierarchies, object properties, data properties and annotations. OWL has three sub languages, which can be ordered by simplicity descending as OWL Lite, OWL-DL (Description Logic), OWL Full (OWL Web Ontology Language Overview, 2004).

Semantic classes and functions

2.1.1 Semantic classes and functions

The text below is a tiny part from OWL file that is generated in the context of the study and describing several data and object properties (see Figure 2.1). It can be read as sentences like below:

- Class name is Building.
- Building class has Floor (Storey) class as component.
- Building has polyhedral surface as geometry type.
- Building has height as float value.

```
# Class: :Building (:Building)
SubClassOf(:Building ObjectSomeValuesFrom(:hasComponent :Floor))
SubClassOf(:Building)
ObjectSomeValuesFrom(:hasGeometryType :PolyhedralSurfaceGeometry))
SubClassOf(:Building DataSomeValuesFrom(:hasHeight xsd:float))
```

Figure 2.1 : Rules defining buildings.

The main purpose of the study is to label building and sub elements fully automatically by using several parameters. Hence, this building ontology model starts from most primitive part: Facet. Facets represents only planar surfaces in any room. Facet class has two data properties: Area and Surface Normal Angle. Both data properties data types are float. Area is created to identify and re-classify smaller parts of walls, ceilings and floors. Surface Normal Angle is used to compare with other angles (like sensor orientation angle or zenith angle in an individual scene) and also these comparisons and calculations can be used for re-classifying.

First subdivision created by angle between surface normal vector and zenith/or nadir axis to split facets into parts: Horizontal, vertical and sloped facet. It is easily predict to it should be used for classify ceiling, floor and walls. Both of ceiling and floors have mostly horizontal geometries, but they can be classified by using angle between surface normal vector and zenith axis, which can be described with following filter that is given in the figure 2.2:

```
(hasSurfaceNormalAngle some  xsd:float[>140f])
and
(hasSurfaceNormalAngle   some  xsd:float[< 220f])
```

Figure 2.2 : Rules defining floors.

Also same filter can be written to ceiling as shown in the figure 2.3.

```
(hasSurfaceNormalAngle some  xsd:float[>-40f])
and
(hasSurfaceNormalAngle some  xsd:float[< 40f])
```

Figure 2.3 : Rules defining ceilings.

Ceiling, base and wall classes have been created as disjoint classes. When individuals registered to these classes, same individuals can not be member of multiple classes at the same time. In other saying, an individual should be a classified by only one of these classes (see Figure 2.4).

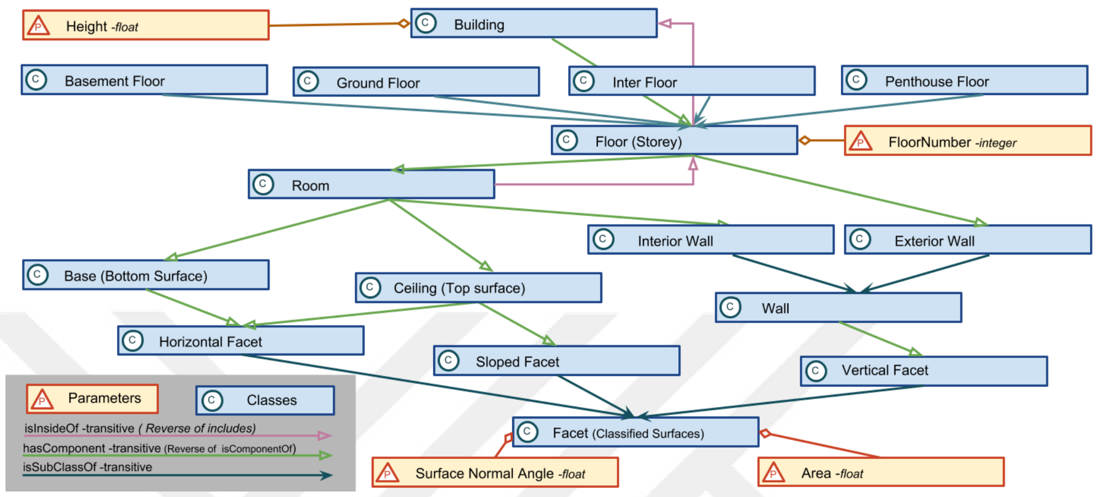


Figure 2.4 : Simplified schematic diagram of semantic labeling.

Wall class has been separated into two classes: Interior Wall and Exterior Wall. The main idea behind this classification is differentiate walls which are observed from inside and outside of building. In most cases, some of interior walls in buildings are also the exterior wall, but in some cases exterior walls are not sharing same volumetric geometry with interior walls. For example, elevator shafts, air wells or some isolation applications are not easily sensible by laser scanners, but they are separate objects which increasing total volume of buildings. The classification between interior wall and exterior wall should be handled by semi-automatically for now by choosing default parameter before scanning survey.

Defining rooms is more complicated than other classes, so it can be said that more complicated rules and definitions preserves more clear definitions of real objects. Here are the defining rules of room:

- hasComponent some (Base and Ceiling and InteriorWall)
    - Room should have at least one base, at least one ceiling and at least one interior wall at the time.
- hasGeometryType some PolyhedralSurfaceGeometry

    - Room geometry should be formed by at least one polyhedral surface.
- hasVolumeTypeByClosure some ClosedVolume
    - Room should be in closed type by volumetric perspective.
- isComponentOf some Floor
    - Room should be a component of at least one floor (storey).
- isInsideOf some Floor
    - Room should be inside of a floor (storey).

Room defining rules designed to excluding some building objects, which are gazebos (no ceiling), gardens (no ceiling), corridors (not closed volume type), greenhouses (no wall), balconies (not in polyhedral surface geometry).

Floor class also shares similar definitions with room class, except several differences: Floor class need FloorNumber parameter in integer data type and floor class formed by base, ceiling and exterior walls. Using exterior walls will be kept shape and volume, so it can be used to create and calculated final shape and volume of building. Floors has been seperated into 4 subclass by floor number: Basement Floor, Ground Floor, Inter Floor and Penthouse Floor.

Finally, floor class formed building class and building class separated into three sub class by total height parameter: Short building, medium building and tall buildings. Height parameter designed as float data type and use only building class as domain.

2.1.2 Value partitions of buildings and sub elements

A user ontology must define objects used by application more specific and clearly rather than other classes in upper ontology (Cotofrei et al., 2011). For more detailed user definitions, user ontologies can be design with more specific classifications and ontology driven method provides many to many relations by using value partitions. In this case, there are four value partitions:

- GeometryType
    - PointGeometry
    - LinestringGeometry
    - PolygonGeometry

    - PolyhedralsurfaceGeometry
- TypeByConstruction
    - Completed
    - UnderConstruction
- TypeByUsage
    - Commercial
    - Industrial
    - Residential
    - InconsistentUsageType
- VolumeTypeByClosure
    - ClosedVolume
    - OpenVolume
    - ConvertibleVolume

Ideally, a user ontology which designed for representing spatial objects should consist three main parts within: Lexicon, Theassauri and Taxonomy. Lexicon is a definition of primitive geometries includes generally dominant planes and simple quadrics (Finat et al., 2010). In this case, most important part of lexicon is dominant planes and it is representing as PolygonGeometry. PointGeometry is used for defining observer or sensor point. LinestringGeometry is also defined to representing cruise path considering robotic unit acts as stop and go measurement method.

Buildings and sub elements ontology model consists in total 38 classes visualized in Figure 2.5, 10 object properties and 4 data properties. Most of sub classes has been defined as disjoint classes, so model can kept individuals registered to only one class in fraternal classes at the same time.

Object properties and their functional characteristics briefly explained in list below:

- hasComponent (Transitive): If a parent class have some child classes as component, than components of child classes would be automatically components of parent classes too. For example, take these axioms as sample: <Huseyin isParentOf Sabri> and <Sabri isParentOf Murat>. If <isParentOf>

property is transitive than this axiom would be also calculated as true: <Huseyin isParentOf Murat>.

- hasGeometryType (Functional): A geometry individual can have only one type at once.
- hasTypeByConstruction (Functional): This property would be used to differ buildings as completed and not completed.
- hasTypeByUsage (Functional): Using to separating buildings to four disjoint subclasses. Only different thing about this value partition is including InconsistentUsageType designed for hold buildings in it till usage type obtained.
- hasVolumeTypeByClosure (Functional): Classifying geometries considering their volume creation ability.
- Includes (Transitive): Checks if an object includes other one spatially. It is inverse property of isInsideOf.
- isComponentOf (Transitive): It is inverse property of hasComponent.
- isInsideOf (Transitive): It is inverse property of includes.

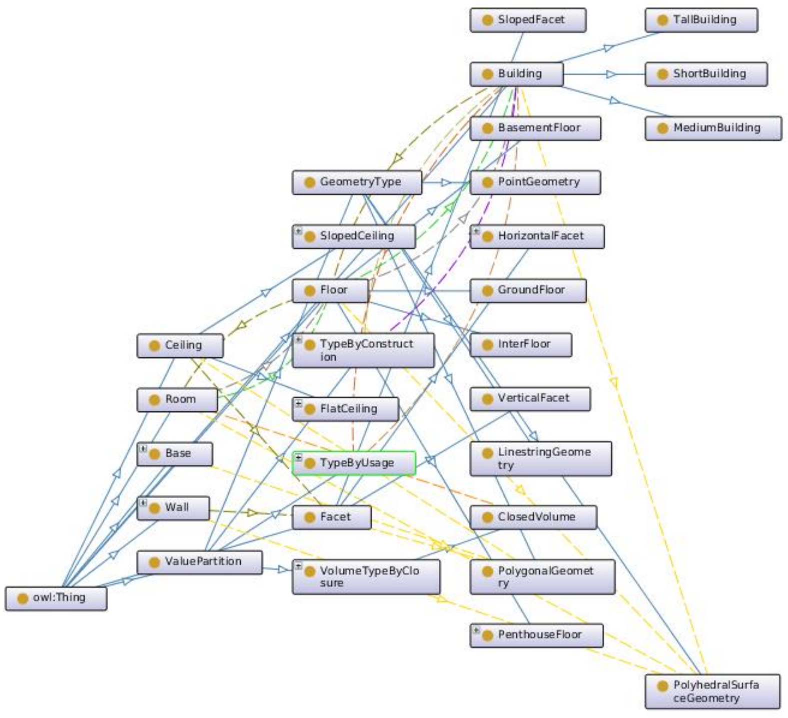


Figure 2.5 : Full schematic of semantic labeling (Every different styled connection lines symbolizing a different object property).

2.2 Point Cloud Processes

Random Sample Consensus (RANSAC) algorithm is a proven iterative method to detect surfaces on both of 2D and 3D space (Schnabel et al., 2008). On the other hand, Hough Transform is also another proven and efficient method for detecting surfaces, but it needs more memory and CPU/GPU consumption on detection progress. Hence, RANSAC method has been chosen for planar surface detection. RANSAC algorithm has been developed firstly in 1981 by Fischler and Bolles at SRI International and used for detecting location of known points with landmarks on an image (Fischler and Bolles, 1981).

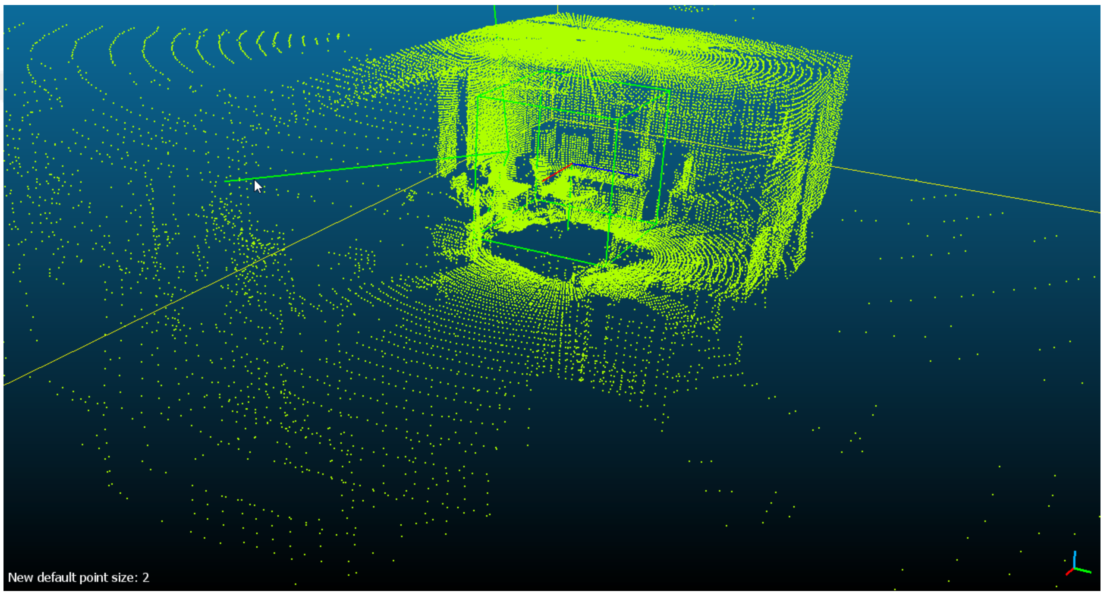


Figure 2.6 : First View of point cloud data.

As it can be easily noticed that original point cloud includes unnecessary points, for instance, reflections, noisy points and measurements observed outside of room because of room gaps, such as windows, doors or etc (see Figure 2.6). In the scope of the study, since it is expected to perform an automatic object recognition as much as possible, the methods used in the study share similar principles with semi-automatically object recognition and classification methods.

A successful object recognition process can be divided into three sub-process (Gallardo et al., 2015):

- Preprocess: Aim to reduce point numbers in a cloud, which would be unnecessary for object recognition

- Descriptor Extraction: Extracts geometric features with mathematical descriptions for every objects
- Classification: Converts geometry features into a meaningful, descriptive information

In the context of the study, Point Cloud Library (PCL) is used to achieve all sub-process described below. PCL is a C++ based, scalable and open library project. PCL includes many functions, supports different data types and it is able to be compiled in most of operating systems. Also, PCL can be implemented with Qt, Eclipse and Visual Studio development frameworks to create graphic user interfaces (GUI) more easily (PCL Tutorials, 2019).

2.2.1 First filter: Statistical Outlier Removal

In the point cloud, which used as test data, PCL has been configured to filter unnecessary points using Statistical Outlier Removal (see Figure 2.7). Statistical Outlier Removal is one of the filtering function in PCL, which is based on principle of gaussian standard deviation and uses distances between points as variable (see Figure 2.8 and 2.9). PCL includes also 39 other classes to filtering point clouds (PCL Documentation, 2018) .

```
  //Filters unnecessary points as outliers in point cloud and assign to a new point
cloud
   pcl::StatisticalOutlierRemoval<pcl::PointXYZ> statFilter;
   statFilter.setInputCloud(cloud);
   statFilter.setMeanK(50);
   statFilter.setStddevMulThresh(1);
   statFilter.filter(*cloud_filtered);
```

Figure 2.7 : Filtering point cloud and storing it using pointer.

setMeanK() is a member function of StatisticalOutlierRemoval class and used for defining number of nearest neighbour points to calculate mean distance between points. setStddevMulThresh() is another member function, used for assigning a standard deviation number for point cloud. This function needs <pcl\filters\statistical_outlier_removal.h> included as header in cpp (default file extension for C++ codes) file.

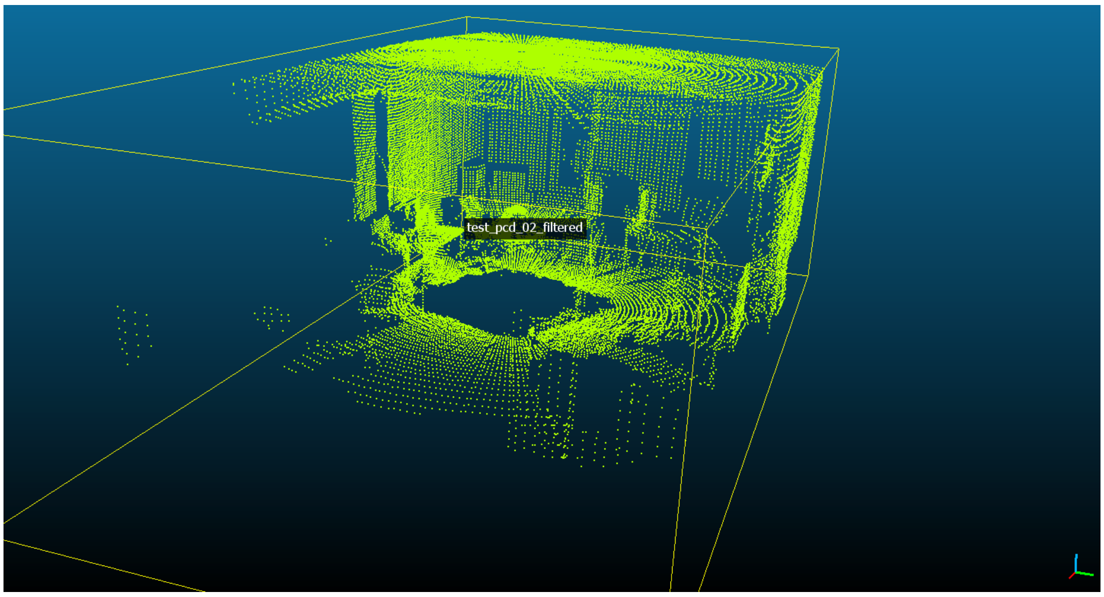


Figure 2.8 : Filtered point cloud data using Statistical Outliers Removal class in PCL.

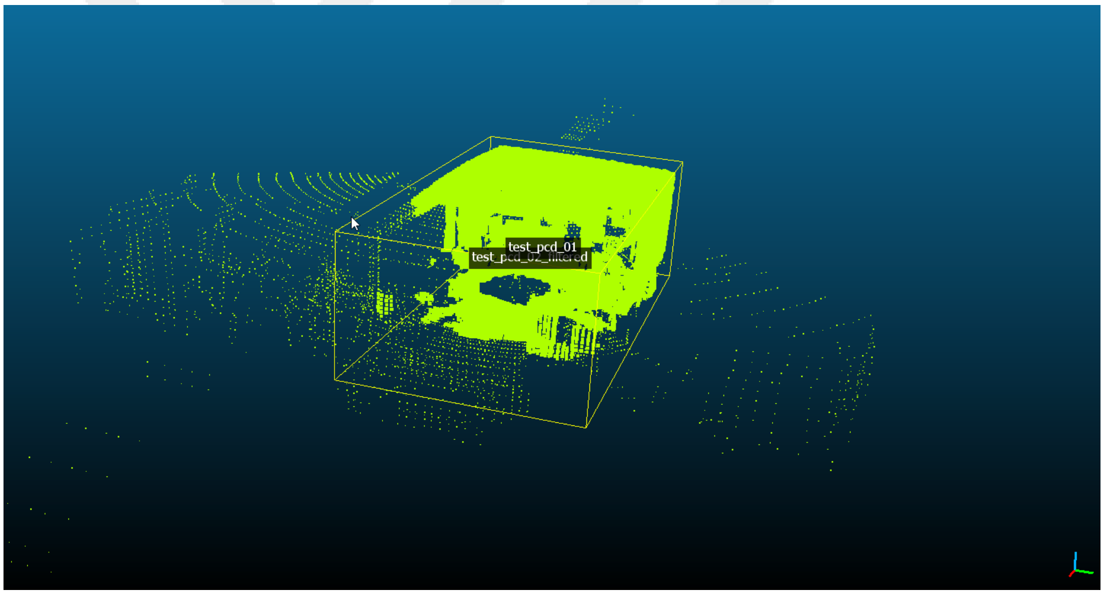
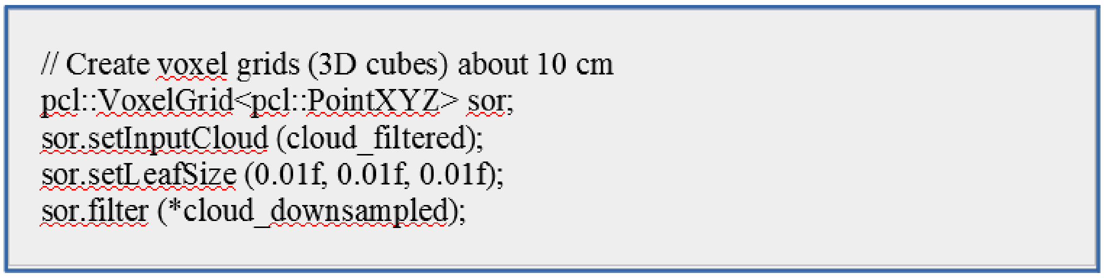


Figure 2.9 : Comparison of point cloud before and after filtered (Smaller sized points are outliers).

2.2.2 Second filter: Voxel Grid

Voxels are 3D cubes contain all represented points in a cloud. As precessor works declare that if voxel size set too large, than multiple shapes could be ignored by recognition functions, in other case if voxel size set too small, than object recognition functions would found more shapes than preferred (Cruz et al., 2007) .



```
// Create voxel grids (3D cubes) about 10 cm
pcl::VoxelGrid<pcl::PointXYZ> sor;
sor.setInputCloud (cloud_filtered);
sor.setLeafSize (0.01f, 0.01f, 0.01f);
sor.filter (*cloud_downsampled);
```

Figure 2.10 : Setting voxel grid filter and save filtered point cloud with pointer.

setLeafSize() function is a member function of VoxelGrid filter class, assigning voxel grid sizes as float numbers in metric unit type (see Figure 2.10 and 2.11). This function needs <pcl\filters\voxel_grid.h> included as header in cpp file.

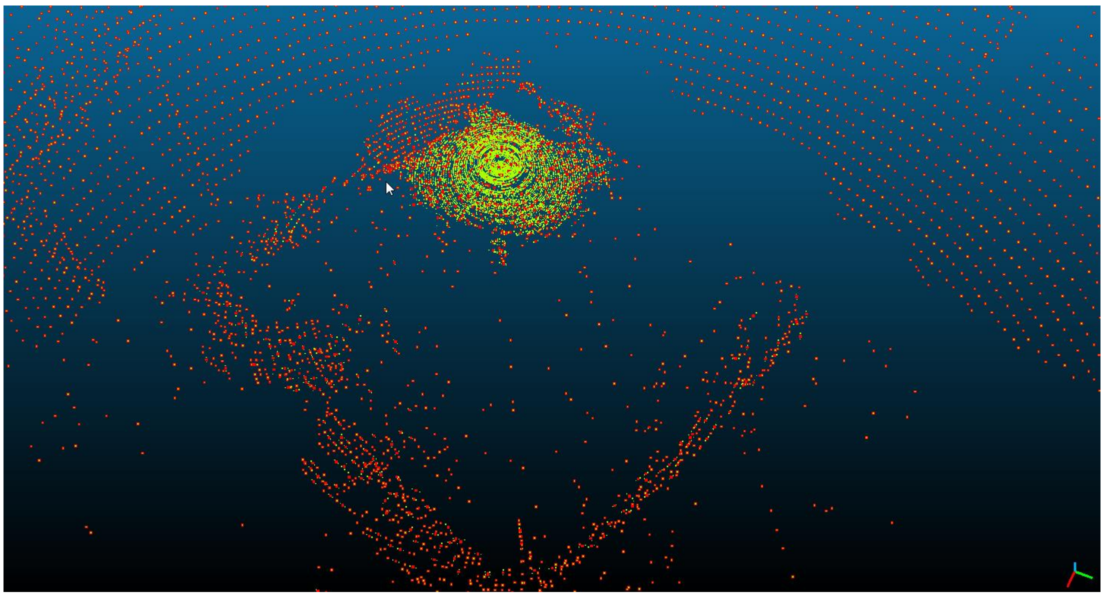


Figure 2.11 : Green points represent first filtered point cloud and red points represent downsampled point cloud using Voxel Grid filter.

2.2.3 RANSAC (RANdom Sample Consensus) segmentation

Random Sample Consensus (RANSAC) algorithm has been introduced firstly at 1981 by SPI International to solve Location Determination Problem (LDP) by determining known points on obtained images (Fischler and Bolles, 1981). This development helped professionals by changing way of detecting objects to an automatic progress and reducing work time on cartography softwares. RANSAC algorithm can be explained by 4 sub-process:

- Select sample data (For exp. Three points in 2D or 3D point cloud)
- Figure an analytic model (For exp. A line, arc or parabolic formula)
- Overlap model with data and count inliers

- Repeat previous function until a confident count number (Confident means sufficient threshold on standard deviation, like %95)

In the same way, RANSAC algorithm can be used for obtain planar surfaces in 3D point clouds (see Figure 2.12, 2.14 and 2.15). In PCL, following classes should be included in main functional code:

- <pcl\sample_consensus\ransac.h>
- <pcl\sample_consensus\sac_model_plane.h>
- <pcl\ModelCoefficients.h>

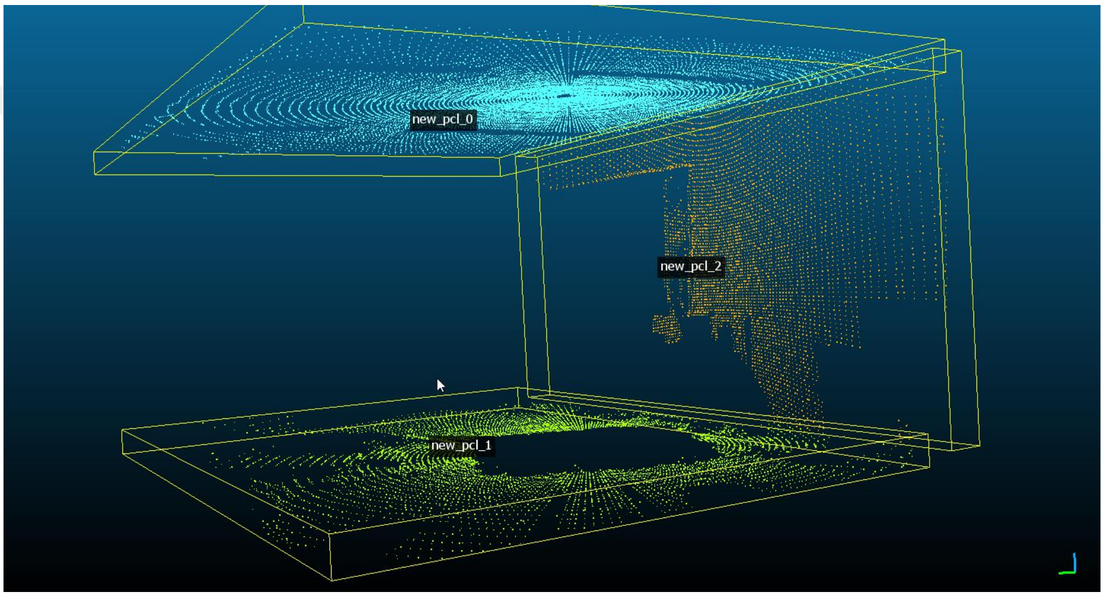


Figure 2.12 : Three sample point cloud subsets after RANSAC Segmentation.

RANSAC is used to only obtain planar surfaces in 3D point cloud with the following code in the figure 2.13 and the code has been explained line by line to avoiding being confused. It can be briefly summarized as following steps:

- Creating necessary structures (ModelCoefficients, PointIndices etc.)
- Defining essential or optional parameters (ModelType, MethodType etc.)
- Creating a loop, which is
    - Setting inliers
    - Saving them
    - Exporting outliers for the next run in loop

At the end of loop function, the point cloud segmented into several parts including only planar surfaces by using %90 of points in the cloud. Different approach and environmental conditions will cause to change this ratio.

```
  //Coefficients of wanted model should be created with structure:
ModelCoefficient

  pcl::ModelCoefficients::Ptr coefficients (new pcl::ModelCoefficients ());

  //Point index numbers in array should be created for inlier points as structure:
PointIndices

  pcl::PointIndices::Ptr inliers (new pcl::PointIndices ());

  //A new point cloud using for SACSegmentation created to only represents X,Y,Z
coordinates.

  //In this step, <pcl::PointXYZRGB> or <pcl::PointXYZI> (w intensity) can be
used

  pcl::SACSegmentation<pcl::PointXYZ> seg;

  //Following setting is optional and increase computation time while outliers are in
majority.

  seg.setOptimizeCoefficients (true);

  seg.setModelType (pcl::SACMODEL_PLANE);

  seg.setMethodType (pcl::SAC_RANSAC);

  seg.setMaxIterations (1000);

  //seg.setEpsAngle( pcl::deg2rad(90.0f) );

  seg.setDistanceThreshold (0.1);

  pcl::ExtractIndices<pcl::PointXYZ> extract;

  int i = 0, nr_points = (int) cloud_downsampled->points.size ();
```

Figure 2.13 : Code block of RANSAC plane detection.

```
while (cloud_downsampled->points.size () > 0.1 * nr_points)

{
// Segment the largest planar component from the remaining cloud

seg.setInputCloud (cloud_downsampled);

seg.segment (*inliers, *coefficients);

if (inliers->indices.size () == 0)

{
std::cerr << "Could not estimate a planar model for the given dataset." << 
std::endl;

Break;
}

 // Extract the inliers

extract.setInputCloud (cloud_downsampled);

extract.setIndices (inliers);

extract.setNegative (false);

extract.filter (*cloud_p);

std::cerr << "PointCloud representing the planar component: " << cloud_p-
>width * cloud_p->height << " data points." << std::endl;

 std::stringstream ss;

int j=i;

//std::string filename = "test_pcd_05_ransacsphere_" + std::to_string(j) + ".pcd";

ss << "new_pcl_" << i << ".pcd";

cout << ss.str () << " " << i << " " << endl;
```

Figure 2.13 : (continue) Code block of RANSAC plane detection.

```
//writer.write<pcl::PointXYZ> (ss.str (), *cloud_p, false);

pcl::copyPointCloud<pcl::PointXYZ>(*cloud_downsampled, *inliers, *cloud_p);

pcl::io::savePCDFileASCII(ss.str () , *cloud_p);

// Create the filtering object

extract.setNegative (true);

extract.filter (*cloud_f);

cloud_downsampled.swap (cloud_f);

i++;

}
```

Figure 2.13 : (continue) Code block of RANSAC plane detection.

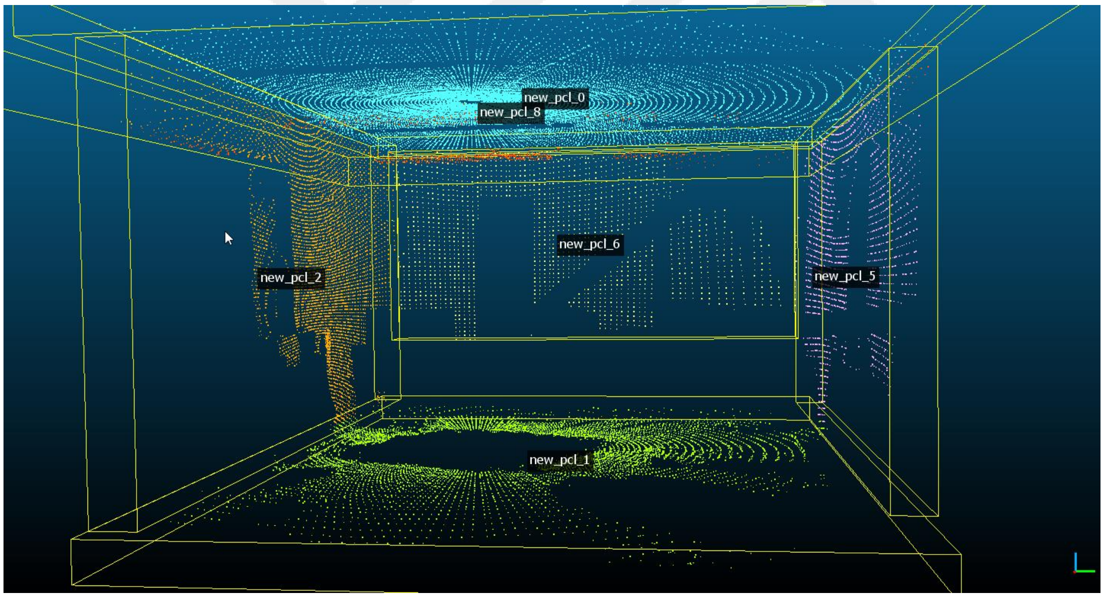


Figure 2.14 : Point clouds extracted at first five iterations from loop function.

2.2.4 Spatial extents of point cloud subsets

PCL Input-Output (PCL IO) library created point cloud subsets by using %91 of all points which filtered by outlier-removal and voxel grid functions (see Figure 2.16). In order to compare point cloud sizes after every progress, firstly source file converted from binary compressed file type to ascii (see Figure 2.17). To achieve this job, there are useful classes in pcd_io library such as, writeASCII, writeBinary, writeBinaryCompressed (PCL Documentation, 2019) . PCDWriter class needs orientation, origin parameters which are able to read from existing point cloud file using Eigen::Point4f point type, and Eigen::Quaternionf data types. Eigen is a template library, which supports linear algebra functions and data types such as vectors, matrices, scalar / vector multiplications and transpositions. So, it is defined with this template class an origin point (Eigen::Point4f - f stands for float and 4 means array size) and a transformation matrice (Eigen::Quaternionf) which is basically represents coefficients of “w + xi + yj + zk” polynomial equation (Eigen Documentation, 2019).

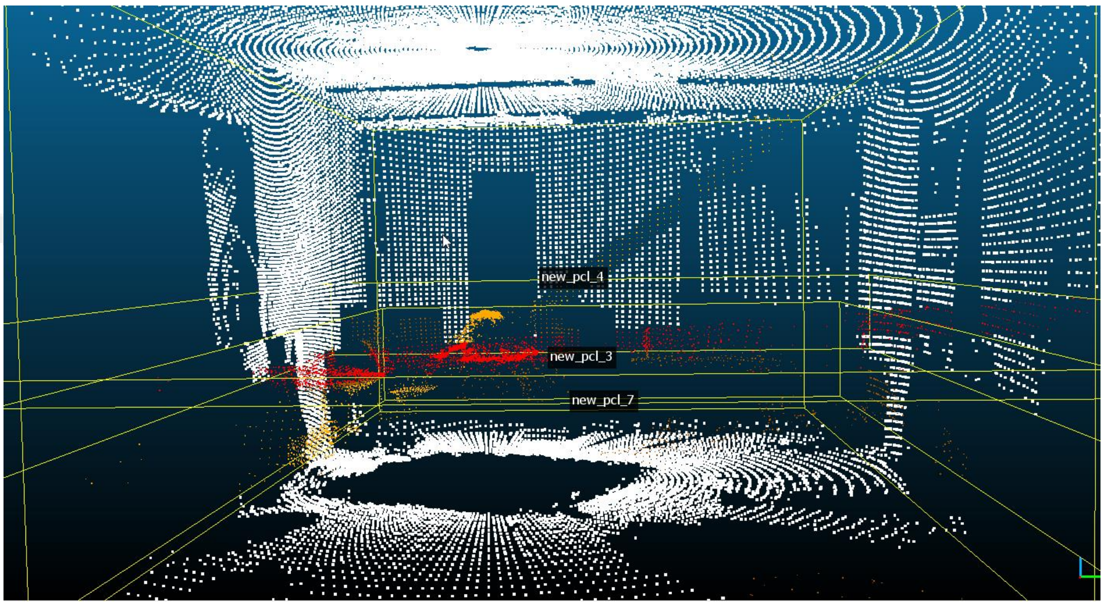
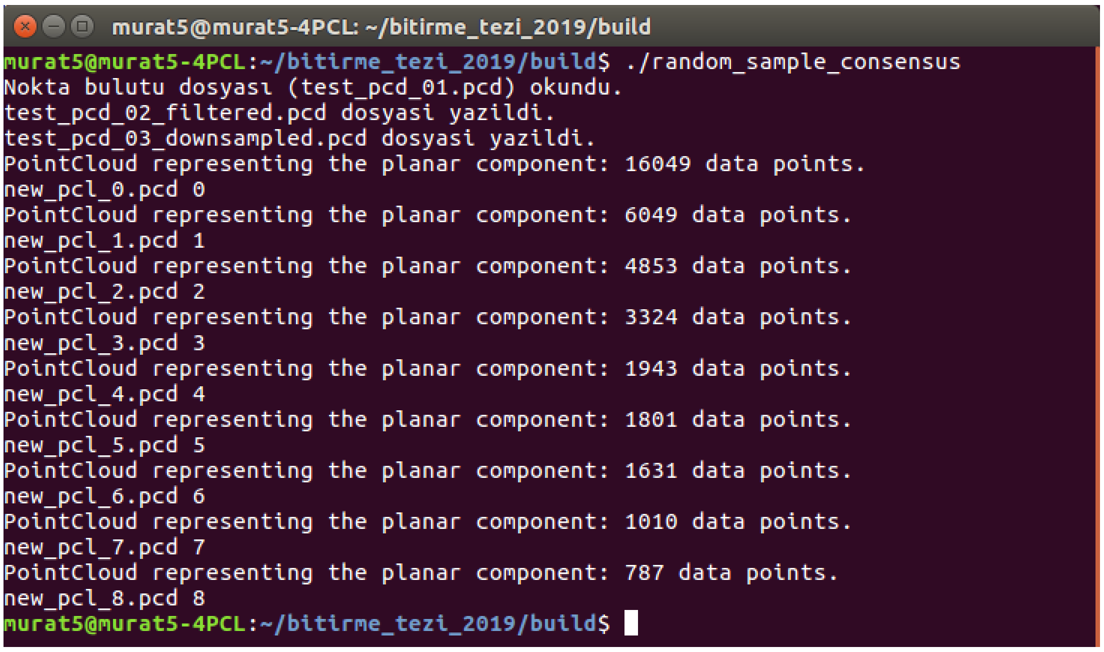


Figure 2.15 : Point clouds extracted from other iterations from loop function.



```
murat5@murat5-4PCL: ~/bitirme_tezi_2019/build
murat5@murat5-4PCL:~/bitirme_tezi_2019/build$ ./random_sample_consensus
Nokta bulutu dosyası (test_pcd_01.pcd) okundu.
test_pcd_02_filtered.pcd dosyasi yazildi.
test_pcd_03_downsampled.pcd dosyasi yazildi.
PointCloud representing the planar component: 16049 data points.
new_pcl_0.pcd 0
PointCloud representing the planar component: 6049 data points.
new_pcl_1.pcd 1
PointCloud representing the planar component: 4853 data points.
new_pcl_2.pcd 2
PointCloud representing the planar component: 3324 data points.
new_pcl_3.pcd 3
PointCloud representing the planar component: 1943 data points.
new_pcl_4.pcd 4
PointCloud representing the planar component: 1801 data points.
new_pcl_5.pcd 5
PointCloud representing the planar component: 1631 data points.
new_pcl_6.pcd 6
PointCloud representing the planar component: 1010 data points.
new_pcl_7.pcd 7
PointCloud representing the planar component: 787 data points.
new_pcl_8.pcd 8
murat5@murat5-4PCL:~/bitirme_tezi_2019/build$ 
```

Figure 2.16 : Command prompt displays exported point cloud subsets.

Following table 2.1 explains point numbers, volume of bounding box volume in default measurement unit and the file size of pcd files after every steps.

Table 2.1 : PCD file sizes and point numbers after filtering and RANSAC Segmentation.

File Name	Application	Point Number	Volume of Bonding Box (m^3)	File Size (KiloBytes)
test_pcd_01	(original file - binary compressed)	112586	1295.55	604
test_pcd_01_ascii	(original file - ASCII)	112586	1295.55	3 432
test_pcd_02_filtered	Statistical Outlier Removal	104196	144.16	3 344
test_pcd_03_downsampled	VoxelGrid	40981	144.16	1 284
RANSAC Plane Segmentation - Point Cloud Subsets				
	Manual Interpretation			
new_pcl_0	(Ceiling)	16049	7.24	492
new_pcl_1	(Floor)	6049	6.86	186
new_pcl_2	(Wall)	4853	4.98	154
new_pcl_3	(Furnite related planes)	3324	27	111
new_pcl_4	(Furnite related planes)	1943	68	67
new_pcl_5	(Wall)	1804	3.96	55
new_pcl_6	(Wall)	1634	2.95	51
new_pcl_7	(Furnite related planes)	1010	25.09	33
new_pcl_8	(Lighting related planes)	787	13.76	24
Total		37453		1173

2.3 Determining Point Clouds Implicit in Walls, Floor and Ceiling Geometries

In previous section, the point clouds, which are strongly belongs to a planar primitive geometry in room, are calculated. In this section, it is needed to determine which planar primitive geometries implicit in walls, ceiling and floor. Testing the angle between normal vector of XY plane and planar primitives is an often used method to handle this problem. As Sanchez tested in his study (Sanchez & Zakhor, 2012), separating point clouds into subsets presents floor and ceiling with 15 degree interval and walls with 45 degree interval is a tested method to determine these objects. Unfortunately, this test data is not a simple empty room, quite the contrary this point cloud data including casual stuffs, blocks some time walls, floor and ceiling. For example, some perpendicular planar surfaces captured from chairs, some parallel to ceiling surfaces captured from lightnings or parallel to floor surfaces captured from tables or other furniture elements.

In this scenario, another method is selected to determine which planar primitives belong to walls, floor and ceiling. It has been decided to calculate a 3D boundary box that presents limits of room after PCL downsample and filtering progress and calculate a buffered volumes created by using 3D faces of boundary box (see Figure 2.18). These buffered volumes are most possible bodies in which walls, floor and ceiling should be contained. After some simple tests, only walls, floor and ceiling geometries can be filtered and saved as 3D polygons or facets, with preferred attributes.

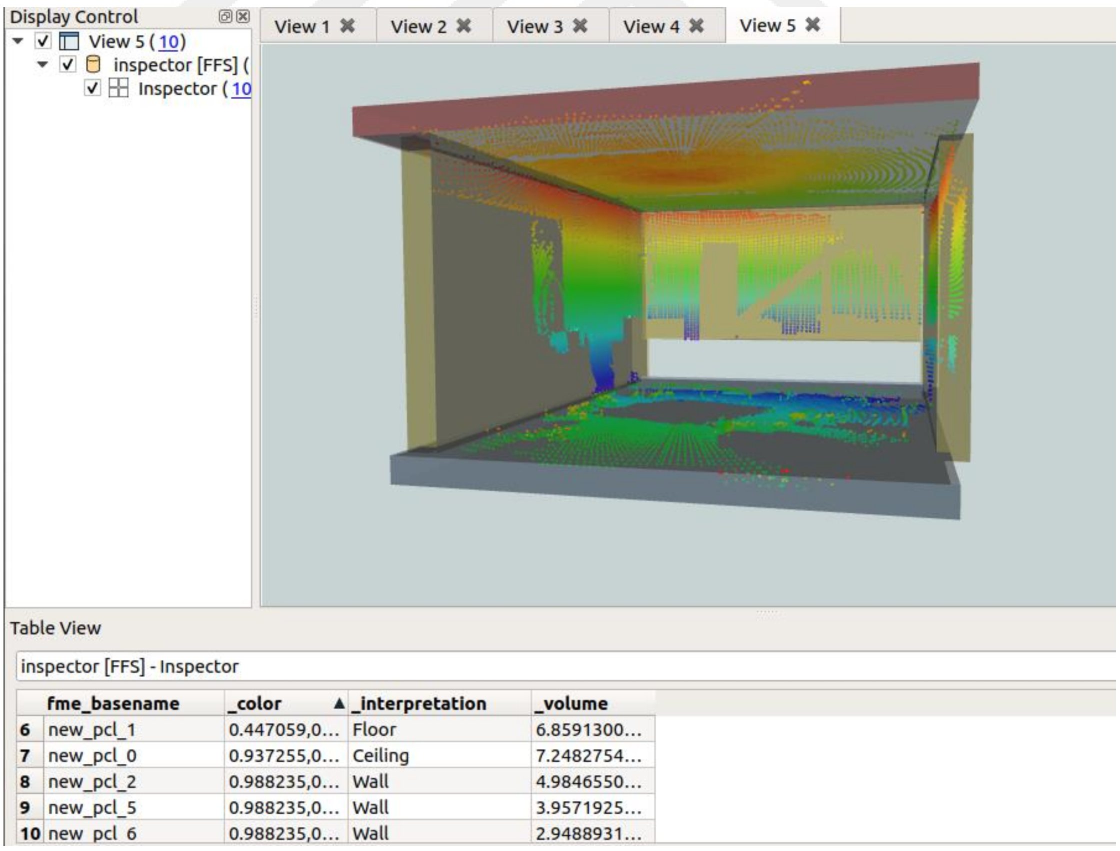


Table View

inspector [FFS] - Inspector

	fme_basename	_color	_interpretation	_volume
6	new_pcl_1	0.447059,0...	Floor	6.8591300...
7	new_pcl_0	0.937255,0...	Ceiling	7.2482754...
8	new_pcl_2	0.988235,0...	Wall	4.9846550...
9	new_pcl_5	0.988235,0...	Wall	3.9571925...
10	new_pcl_6	0.988235,0...	Wall	2.9488931...

Figure 2.17 : Point cloud (PCD) files and individual boundary boxes of walls, floor and ceiling geometries filtered manually and visualized by FME Data Inspector.

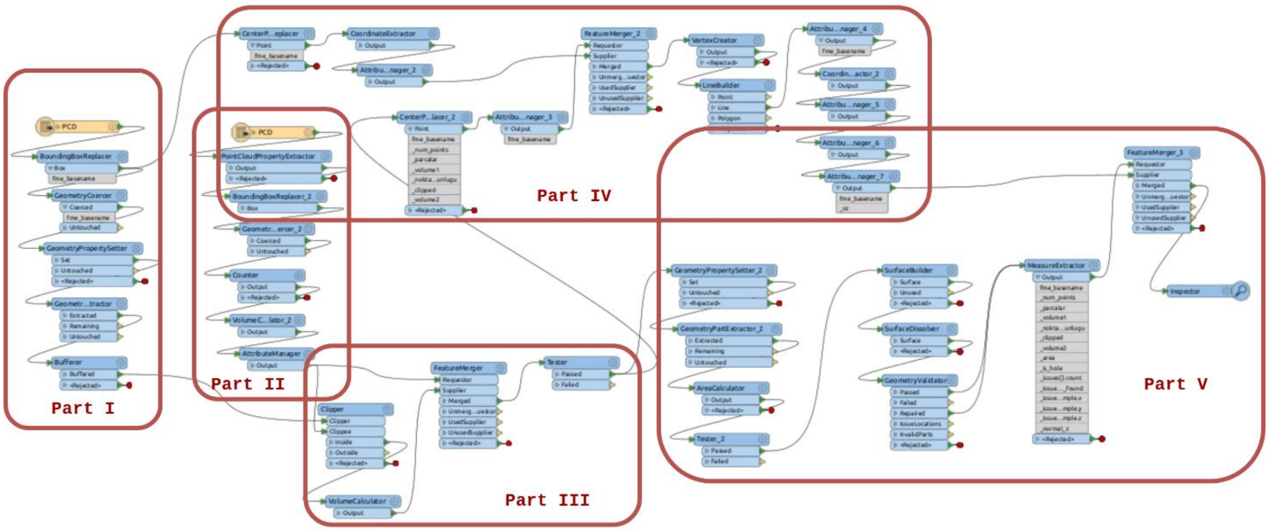


Figure 2.18 : FME based progress determines which PCD files belongs to walls, floor and ceiling object.

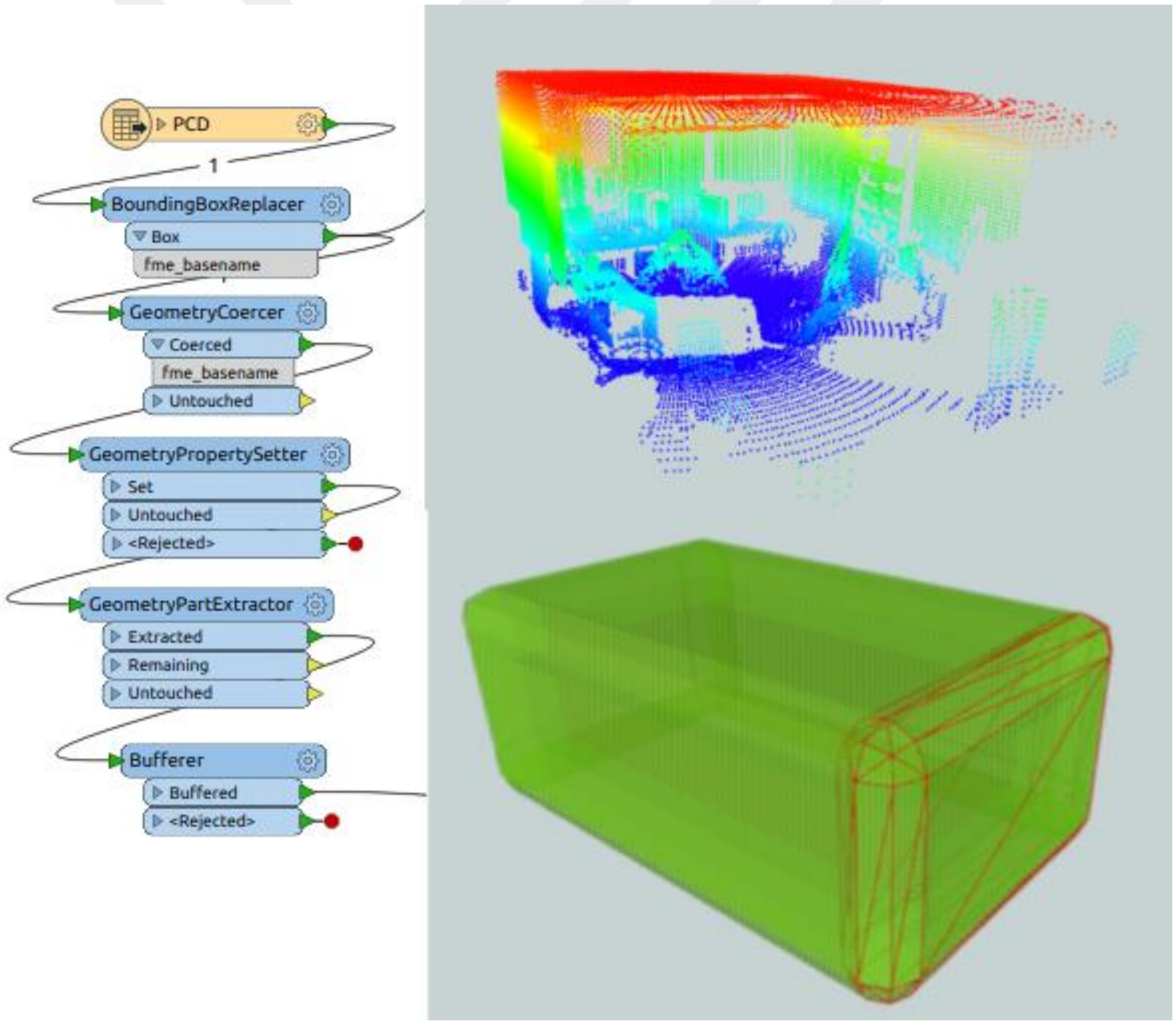


Figure 2.19 : Visualizes all used transformers (functions) to calculate and recreate buffer volumes for walls, floor and ceiling.

2.3.1 Buffering bounding box of whole point cloud

First progress has been used to calculate volumes in which walls, floor and ceiling should be contained. The resampled point cloud has replaced by 3D bounding box, than all faces of boundary box extracted as 3D faces. All 3D faces used as a base surface to calculate a buffered volume within 0.6 meters thickness (see Figure 2.19 and Table 2.2).

Table 2.2 : Description of used transformers to calculate and recreate buffer volumes.

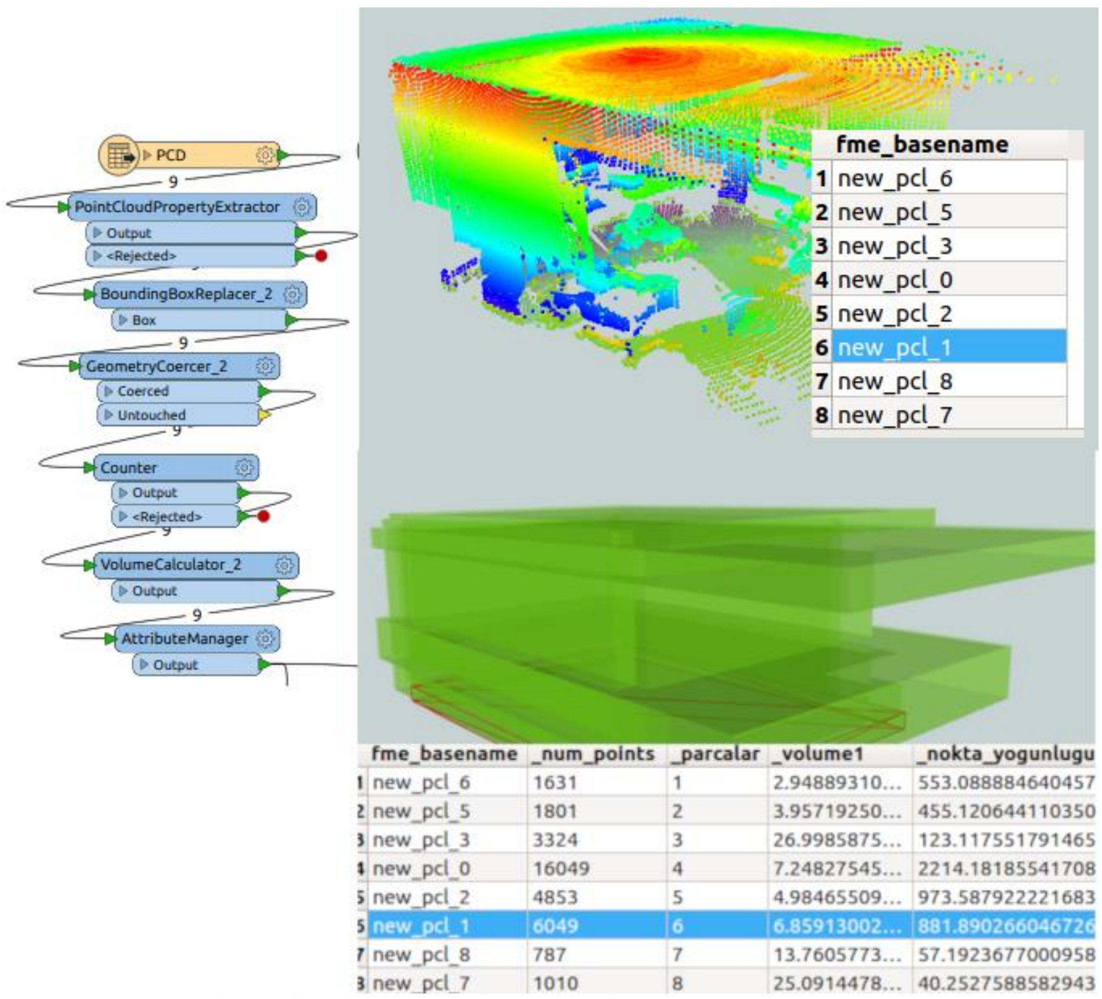
PCD (File Reader)	Test_pcd_03_downsampled.pcd file is read by FME library.
BoundingBoxReplacer	"3D bounding cube" option selected. Output ports divided into two different progress, one of them used to calculate buffered volumes and other one used to calculate center points and vectors.
GeometryCoercer	While FME supports different geometry types, IFMEBox (3D Box) type geometry converted to IFMEBREPSolid (3D Solid) geometry.
GeometryPropertySetter	This function counted IFMEFace geometries belongs to solid geometry.
GeometryPartExtractor	Face geometries extracted as features by this function.
Bufferer	Every 3D face converted to a volume created by 0.6 meter buffer parameter.

2.3.2 Calculating properties of point clouds created by RANSAC segmentation

Second progress has read all point cloud files created by RANSAC algorithm and calculate point numbers in every cloud. Point number was used to calculate point density in every bounding boxes (see Figure 2.20 and Table 2.3).

2.3.3 Testing intersections of buffered volumes and point cloud bounding boxes

FME has some functionalities to make boolean operations like intersect, difference and union, just like in 2D geometries. Clipper is one of these functions and is capable to reproduce intersections and difference volumes with considering clipper or clipee priority. After this operation, intersected volumes has been calculated and first given solids filtered using intersected / not intersected volumes with "tester" function (see Figure 2.21, 2.22 and Table 2.4).



	fme_basename
1	new_pcl_6
2	new_pcl_5
3	new_pcl_3
4	new_pcl_0
5	new_pcl_2
6	new_pcl_1
7	new_pcl_8
8	new_pcl_7

	fme_basename	_num_points	_parcalar	_volume1	_nokta_yogunlugu
1	new_pcl_6	1631	1	2.94889310...	553.088884640457
2	new_pcl_5	1801	2	3.95719250...	455.120644110350
3	new_pcl_3	3324	3	26.9985875...	123.117551791465
4	new_pcl_0	16049	4	7.24827545...	2214.18185541708
5	new_pcl_2	4853	5	4.98465509...	973.587922221683
6	new_pcl_1	6049	6	6.85913002...	881.890266046726
7	new_pcl_8	787	7	13.7605773...	57.1923677000958
8	new_pcl_7	1010	8	25.0914478...	40.2527588582943

Figure 2.20 : Visualizes transformers to derive point cloud properties and point cloud bounding boxes.

Table 2.3 : Description of used transformers to derive point cloud properties and point cloud bounding boxes

PCD (File Reader)	All PCD files are read, which are exported with PCL RANSAC algorithm.
PointCloudPropertyExtractor	Point number calculated and stored as "_num_points" attribute.
BoundingBoxReplacer_2	3D bounding boxes are created.
GeometryCoercer_2	All box features converted to BREP solid features.
Counter	Features are counted and stored as "parcalar" attribute.
VolumeCalculator_2	Volumes of solids are calculated.
AttributeManager	Point Densities are calculated by dividing point number to volume and stored as "_nokta_yogunlugu" attribute.

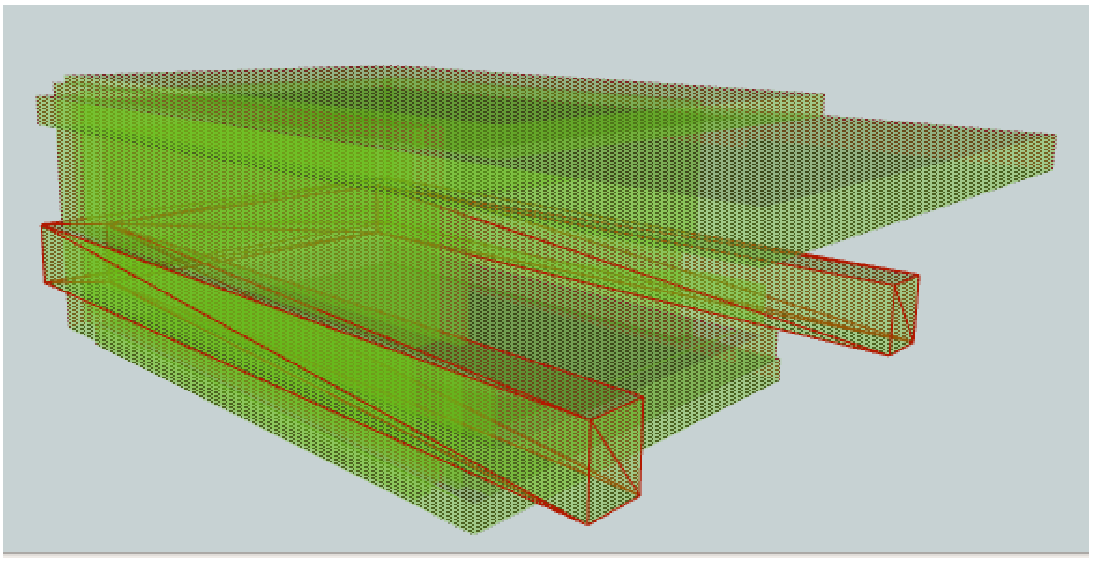
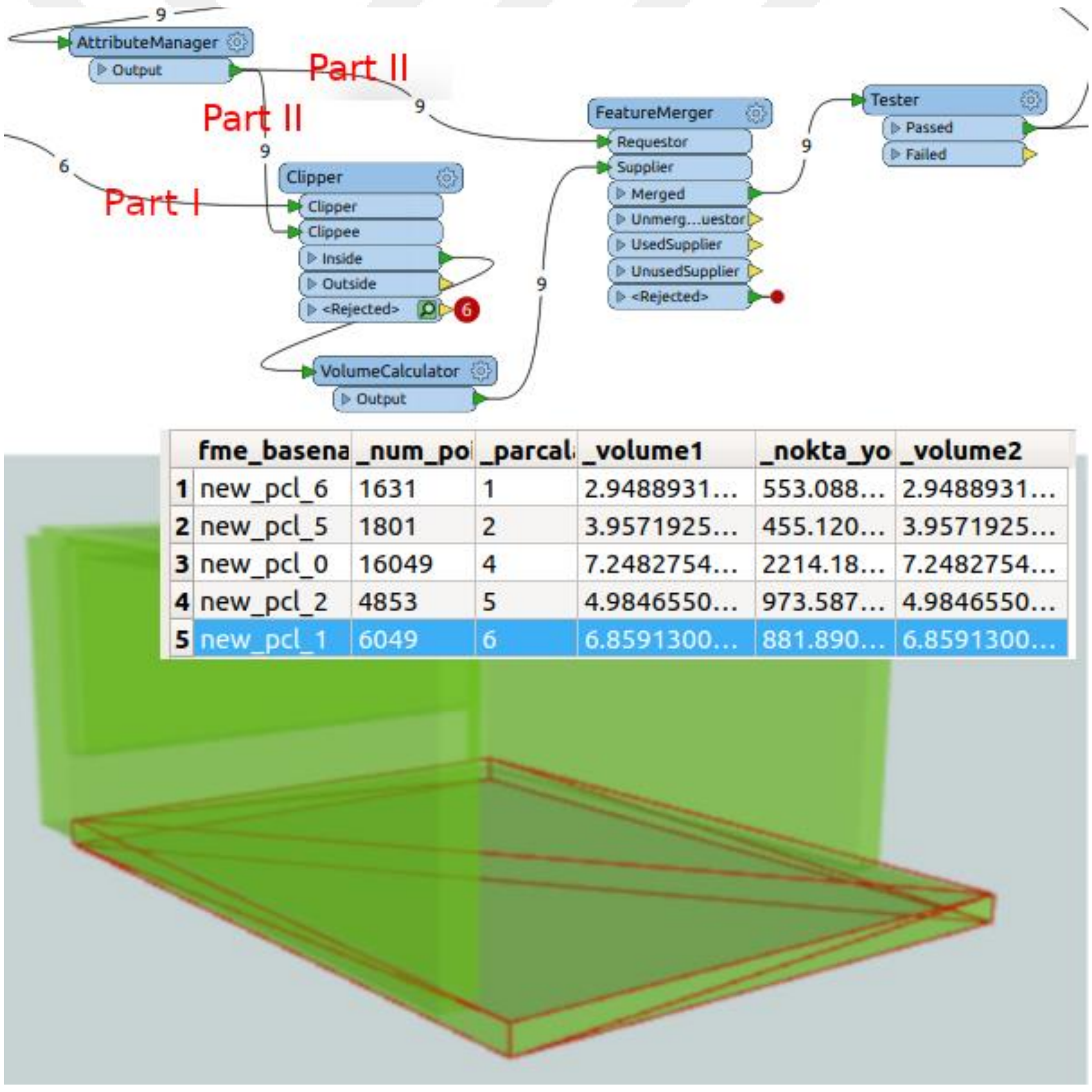


Figure 2.21 : Intersected volumes between point clouds extracted by RANSAC and buffer volume of whole point cloud.



	fme_basena	_num_poi	_parcal	_volume1	_nokta_yo	_volume2
1	new_pcl_6	1631	1	2.9488931...	553.088...	2.9488931...
2	new_pcl_5	1801	2	3.9571925...	455.120...	3.9571925...
3	new_pcl_0	16049	4	7.2482754...	2214.18...	7.2482754...
4	new_pcl_2	4853	5	4.9846550...	973.587...	4.9846550...
5	new_pcl_1	6049	6	6.8591300...	881.890...	6.8591300...

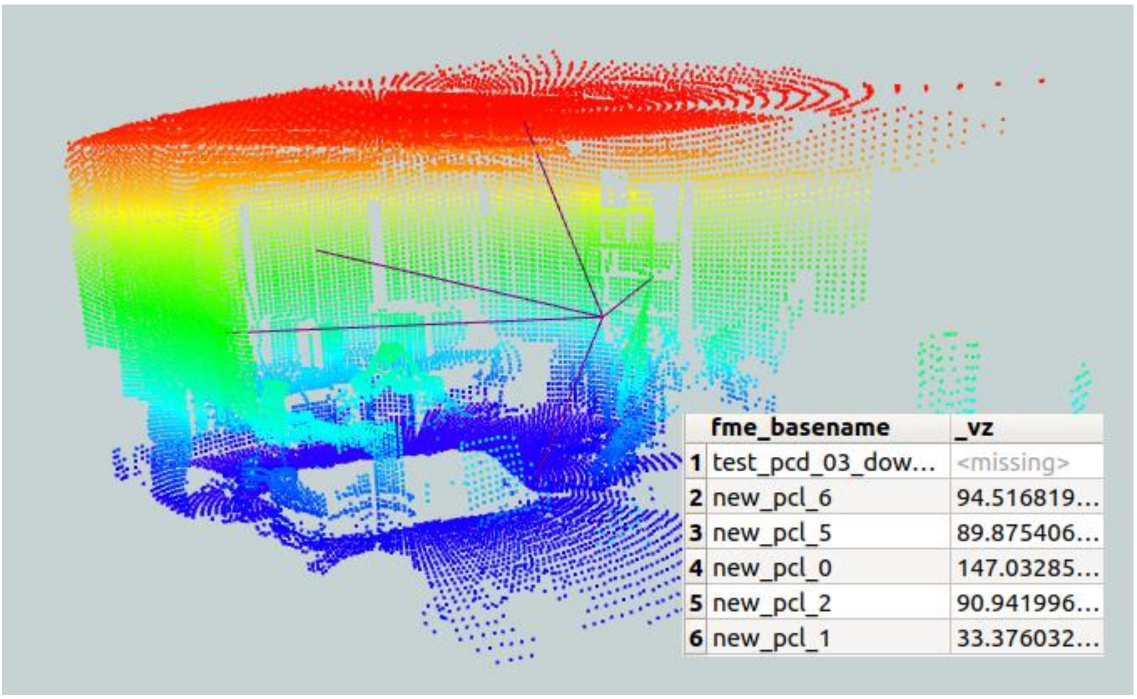
Figure 2.22 : Visualizes intersection, calculation volume of intersected parts and test progress.

Table 2.4 : Description of transformers to intersection and testing volumes

Clipper	"Treat as Inside" option selected. Buffered solids used as "Clipper" input and point cloud boundary boxes used as "Clipee" input. Only "Inside" output used to send to next transformer.
VolumeCalculator_2	Volumes of intersections calculated.
FeatureMerger	Point Cloud data coming from (Part II) is merged with only attributes of calculated intersection volumes.
Tester	This function tests following conditions: 1. Point Density > 400 point/m3 2. Intersected Volume / First Volume > %90

2.3.4 Creating vectors from surface center points to the gravity point of room

Extracting normal vectors of planar primitives is a simple progress in PCL, on the other hand a different method has been chosen, reproducing bounding boxes for whole room and also for every planar primitive objects and selecting walls, floor and ceiling objects according to relation with room. Hence, in this section vectors were created between surface center point and gravity center point of room. Such an approach provides to filter objects according to angles of vector components (see Figure 2.23, 2.24 and Table 2.5).



	fme_basename	_vz
1	test_pcd_03_dow...	<missing>
2	new_pcl_6	94.516819...
3	new_pcl_5	89.875406...
4	new_pcl_0	147.03285...
5	new_pcl_2	90.941996...
6	new_pcl_1	33.376032...

Figure 2.23 : Visualizes vector lines and their angular component according to Z axis.

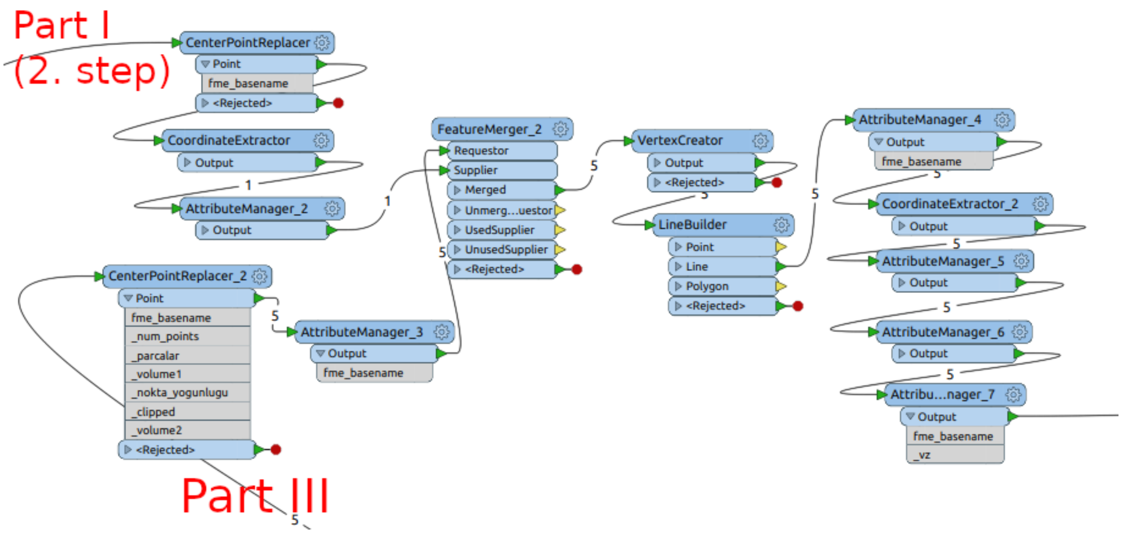
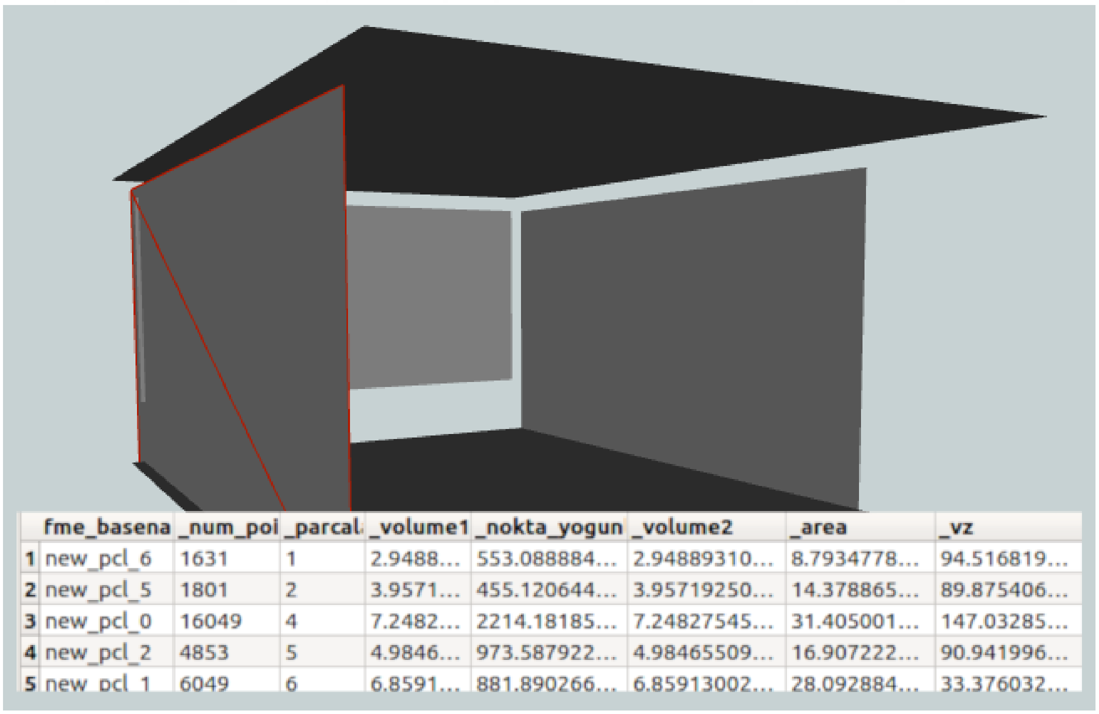


Figure 2.24 : Transformers used to calculate and redraw vector lines

2.3.5 Determining walls, floor and ceiling and store with preferred format

As mentioned before, it is always needed to have values of geometries in order to automatically classify surfaces as walls, floors and ceiling. In this situation, "_vz" which represents angle between vector and Z axis is most important value to prepare an algorithm (see Figure 2.25, 2.26 and Table 2.6). Other attributes, like area, total volume and point density could also be used to check this classification.



	fme_basena	_num_poi	_parcal	_volume1	_nokta_yogun	_volume2	_area	_vz
1	new_pcl_6	1631	1	2.9488...	553.088884...	2.94889310...	8.7934778...	94.516819...
2	new_pcl_5	1801	2	3.9571...	455.120644...	3.95719250...	14.378865...	89.875406...
3	new_pcl_0	16049	4	7.2482...	2214.18185...	7.24827545...	31.405001...	147.03285...
4	new_pcl_2	4853	5	4.9846...	973.587922...	4.98465509...	16.907222...	90.941996...
5	new_pcl_1	6049	6	6.8591...	881.890266...	6.85913002...	28.092884...	33.376032...

Figure 2.25 : Shows polygons which are created based on bounding boxes

Table 2.5 : Transfomers used to calculate vector lines and its angular component according to Z axis

CenterPointReplacer	Bounding box of room replaced with gravity center point (type: 3D point).
CoordinateExtractor	Coordinates of 3D point extracted and stored in "_indices" array.
AttributeManager_2	Just to remove fme_basename attribute
CenterPointReplacer_2	Bounding boxes of wall, floor or ceiling candidates replaced with center point.
AttributeManager_3	Just used to remove unnecessary attributes.
FeatureMerger_2	This transformer creates point bundles, so every center point of wall, floor or ceiling surfaces pairs with gravity center point of room.
VertexCreator	"_indices" array converted to vertexes, so the feature data type converted automatically to MultiPoint.
LineBuilder	Multipoint features converted to lines.
AttributeManager_4	Unnecessary attributes removed.
CoordinateExtractor_2	Now all coordinates of lines are stored in one two dimensional array.
AttributeManager_5	Vector length calculated using following formula and stored as "_birlesik_vektor": sqrt(@pow((@Value(_indices{1}.x)-@Value(_indices{0}.x)),2)+@pow((@Value(_indices{1}.y)-@Value(_indices{0}.y)),2)+@pow((@Value(_indices{1}.z)-@Value(_indices{0}.z)),2)) Represents 3D vector length formula: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$
AttributeManager_6	Angle between Z axis and vectors calculated with following formula and stored as "_vz": @radToDeg(@acos((@Value(_indices{1}.z)-@Value(_indices{0}.z))/@Value(_birlesik_vektor))) Represents inverse cosine formula: $cos^{-1}(\frac{z_2 - z_1}{vector\ length})$
AttributeManager_7	Unnecessary attributes removed.

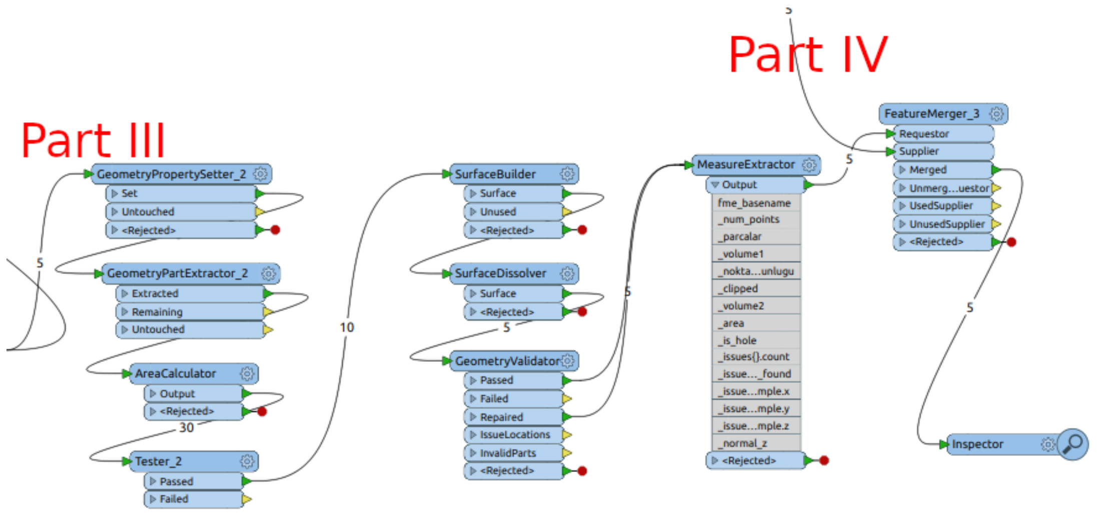


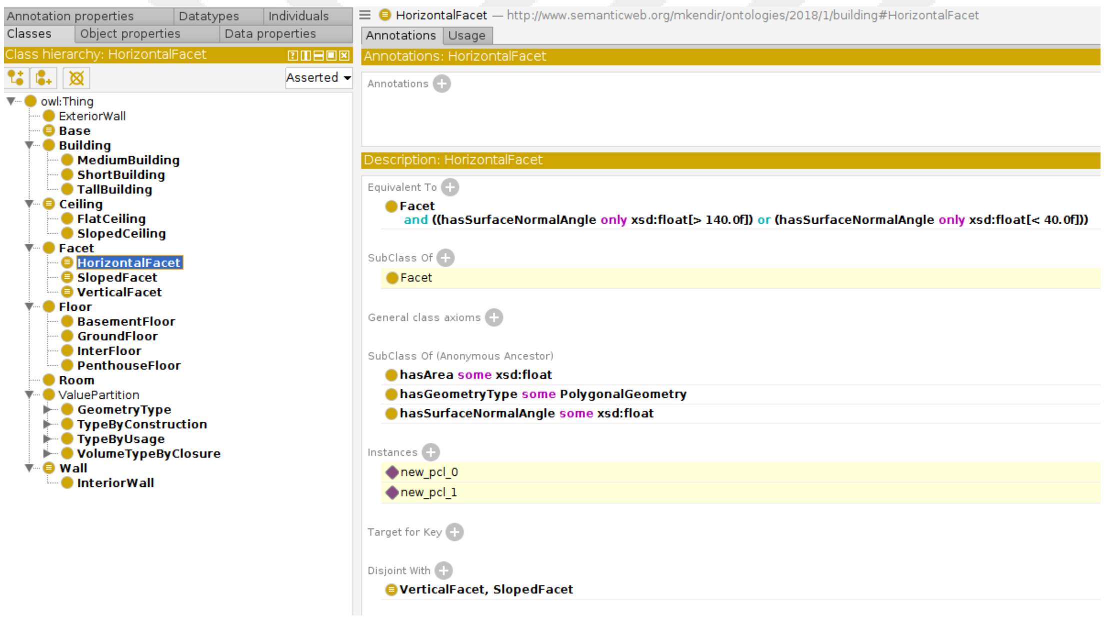
Figure 2.26 : Transformers used to create polygons based on filtered bounding boxes

Table 2.6 : Transformers and descriptions used to create polygons and their attributes

GeometryPropertySetter_2	This transformer count faces of boundary boxes of filtered point clouds.
GeometryPartExtractor_2	All faces of boundary boxes extracted as features.
AreaCalculator	Area of extracted faces calculated and stored as "_area" attribute.
Tester_2	This filter eliminates narrow surfaces of boundary boxes testing volume to area ratio.
SurfaceBuilder	3D polygon features converted to surface features.
SurfaceDissolver	Surfaces with same attributes, which also created from same point clouds dissolved as one. It can be said that boundary box of point clouds converted to an averaged surface.
GeometryValidator	Checks if vertex normals missing.
MeasureExtractor	Checks normal_z value.
FeatureMerger	This transformer merged features with "_vz" attribute represents z angular component of vector.

2.3.6 Importing polygonal geometries into Protege using OWL-DL

FME calculated all essential attributes and simplified geometries using WKT format. WKT format is a very common method to save geometries and transfer them to any kind of spatial data environment. On the other hand, there is not any spatial representation tool/extension in Protege software for geometries. It is decided to store them only for interpretation. Final products are saved in CSV format and every rows in CSV based table has been imported to Protege software. There are several methods to import table rows into Protege as individuals. One of them is manually creating individuals with using data properties and object properties one by one. Other method is connecting to databases using Data Master plugin, which is compatible with Protege-OWL 3.4 version and supports relational database and excel documents. It is obvious thing, that database connection with semantic reasoning is a powerful tool which creates limitless possibilities to create axioms, but nowadays Data Master Plugin and Protege-OWL 3.4 version is out-of-date and have another successor called: Cellfie.



Class hierarchy: HorizontalFacet

```
owl:Thing
    ExteriorWall
    Base
    Building
        MediumBuilding
        ShortBuilding
        TallBuilding
    Ceiling
        FlatCeiling
        SlopedCeiling
    Facet
        HorizontalFacet
        SlopedFacet
        VerticalFacet
    Floor
        BasementFloor
        GroundFloor
        InterFloor
        PenthouseFloor
    Room
    ValuePartition
        GeometryType
        TypeByConstruction
        TypeByUsage
        VolumeTypeByClosure
    Wall
        InteriorWall
```

HorizontalFacet — http://www.semanticweb.org/mkendir/ontologies/2018/1/building#HorizontalFacet

Annotations: HorizontalFacet

Description: HorizontalFacet

Equivalent To
Facet
and ((hasSurfaceNormalAngle only xsd:float[> 140.0f]) or (hasSurfaceNormalAngle only xsd:float[< 40.0f]))

SubClass Of
Facet

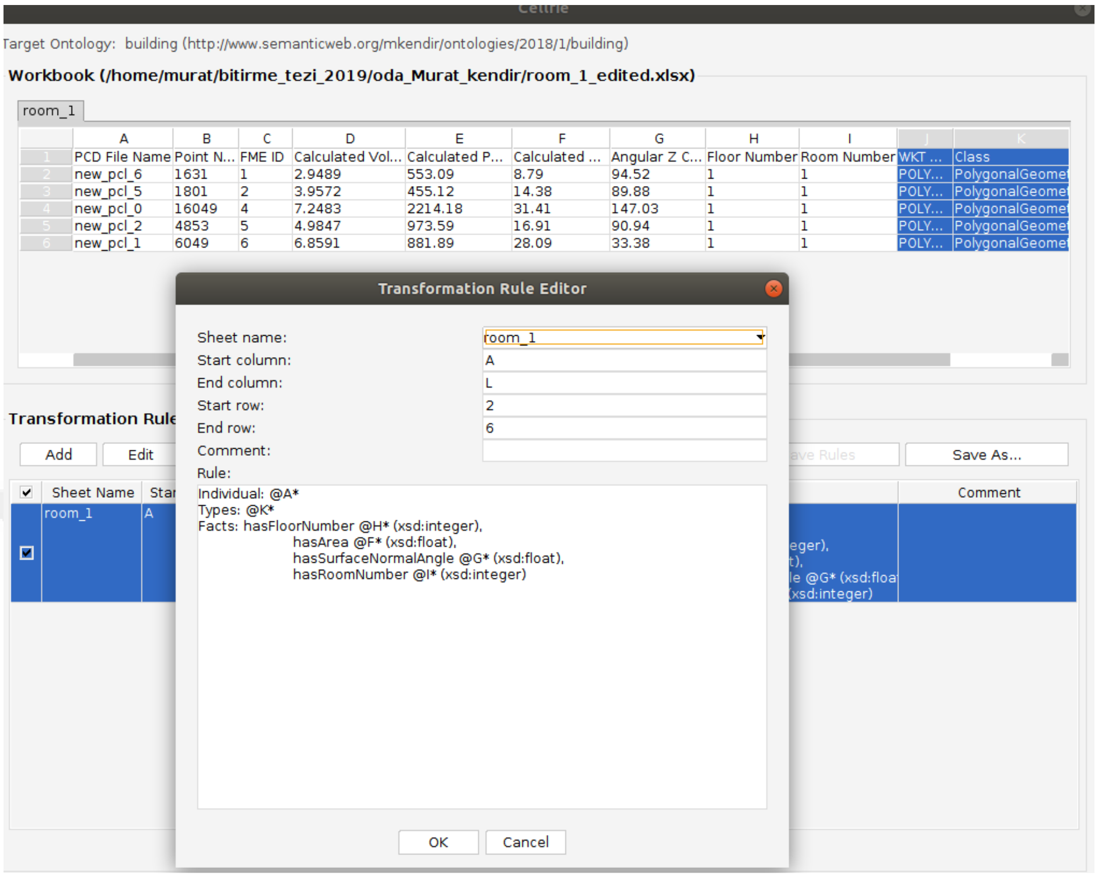
SubClass Of (Anonymous Ancestor)
hasArea some xsd:float
hasGeometryType some PolygonalGeometry
hasSurfaceNormalAngle some xsd:float

Instances
new_pcl_0
new_pcl_1

Disjoint With
VerticalFacet, SlopedFacet

Figure 2.27 : Imported individuals populated ontology classes automatically with OWL-DL rules.

Cellfie is a tool to create any kind of axiom multiply and embedded directly with the Protege software. It is accessible under Tool menu by clicking “Create axioms from Excel workbook”. In this tool, table structured data can be converted to axioms automatically by rules (see Figure 2.27). Default file format for table structured data is XSLX (Microsoft Office 2013-2019 format), but CSV (Comma Seperated Values, an open format for data storage) files can be also used. Using multiple rules at once for one table is easily adaptable or it is possible to use multiple expressions to create multiple axioms in one rule. In this study, Class and other axioms which refers to data properties are created by one rule (See Figure 2.28).



Target Ontology: building (http://www.semanticweb.org/mkendir/ontologies/2018/1/building)

Workbook (/home/murat/bitirme_tezi_2019/oda_Murat_kendir/room_1_edited.xlsx)

	A	B	C	D	E	F	G	H	I	J	K
1	PCD File Name	Point N...	FME ID	Calculated Vol...	Calculated P...	Calculated ...	Angular Z C...	Floor Number	Room Number	WKT ...	Class
2	new_pcl_6	1631	1	2.9489	553.09	8.79	94.52	1	1	POLY...	PolygonalGeomet
3	new_pcl_5	1801	2	3.9572	455.12	14.38	89.88	1	1	POLY...	PolygonalGeomet
4	new_pcl_0	16049	4	7.2483	2214.18	31.41	147.03	1	1	POLY...	PolygonalGeomet
5	new_pcl_2	4853	5	4.9847	973.59	16.91	90.94	1	1	POLY...	PolygonalGeomet
6	new_pcl_1	6049	6	6.8591	881.89	28.09	33.38	1	1	POLY...	PolygonalGeomet

Transformation Rule

Transformation Rule Editor

Sheet name:	room_1
Start column:	A
End column:	L
Start row:	2
End row:	6
Comment:	

Rule:

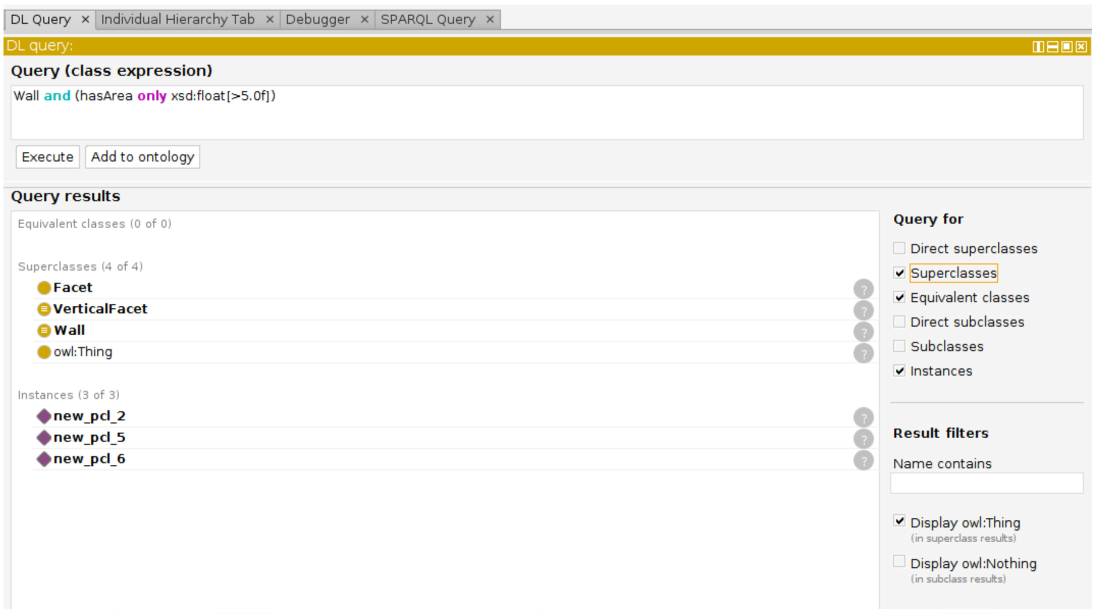
```
Individual: @A*
Types: @K*
Facts: hasFloorNumber @H* (xsd:integer),
               hasArea @F* (xsd:float),
               hasSurfaceNormalAngle @G* (xsd:float),
               hasRoomNumber @I* (xsd:integer)
```

Figure 2.28 : Automatically importing individuals with their data and object properties to Protege using Cellfie tool.

2.3.7 Query the semantic data by DL Query and SPARQL

Querying the data in Protege is possible in two ways with embedded tools: DL Query and SPARQL Query. DL Query is easier to handle simple queries because of some functions in user interface(see Figure 2.29).

With the SPARQL whole created OWL and additional semantic domains can be queried at once. Following code has been run directly in the building ontology after individuals imported (see Figure 2.30).



Query (class expression)

```
Wall and (hasArea only xsd:float[>5.0f])
```

Query results

Equivalent classes (0 of 0)

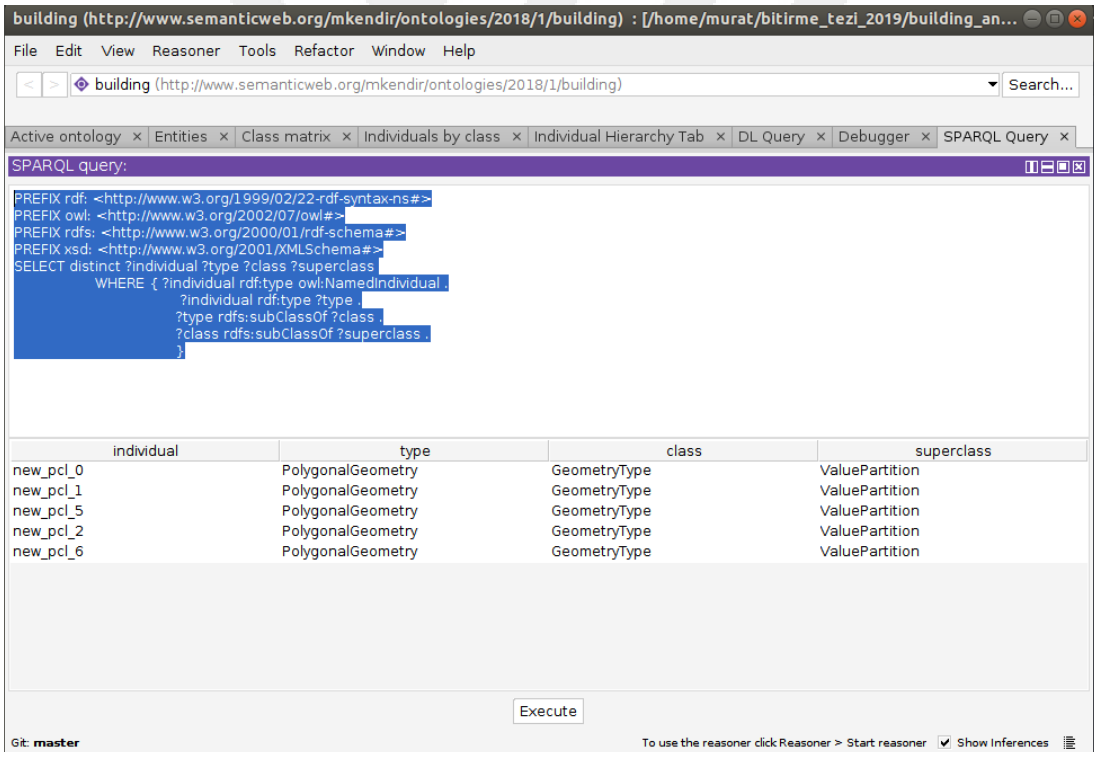
Superclasses (4 of 4)
- Facet
- VerticalFacet
- Wall
- owl:Thing

Instances (3 of 3)
- new_pcl_2
- new_pcl_5
- new_pcl_6

Query for
- [ ] Direct superclasses
- [x] Superclasses
- [x] Equivalent classes
- [ ] Direct subclasses
- [ ] Subclasses
- [x] Instances

- [x] Display owl:Thing (in superclass results)
- [ ] Display owl:Nothing (in subclass results)

Figure 2.29 : DL Query Syntax is same with other expressions in Protege.


building (http://www.semanticweb.org/mkendir/ontologies/2018/1/building) : [/home/murat/bitirme_tezi_2019/building_an...
File Edit View Reasoner Tools Refactor Window Help
building (http://www.semanticweb.org/mkendir/ontologies/2018/1/building)
Search...
Active ontology
Entities
Class matrix
Individuals by class
Individual Hierarchy Tab
DL Query
Debugger
SPARQL Query
SPARQL query:
Execute
Git: master
To use the reasoner click Reasoner > Start reasoner
Show Inferences

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT distinct ?individual ?type ?class ?superclass
          WHERE { ?individual rdf:type owl:NamedIndividual .
                    ?individual rdf:type ?type .
                    ?type rdfs:subClassOf ?class .
                    ?class rdfs:subClassOf ?superclass .
                    }
```

individual	type	class	superclass
new_pcl_0	PolygonalGeometry	GeometryType	ValuePartition
new_pcl_1	PolygonalGeometry	GeometryType	ValuePartition
new_pcl_5	PolygonalGeometry	GeometryType	ValuePartition
new_pcl_2	PolygonalGeometry	GeometryType	ValuePartition
new_pcl_6	PolygonalGeometry	GeometryType	ValuePartition

Figure 2.30 : SPARQL query sample selects individuals with classes and super classes.

3. CONCLUSIONS AND RECOMMENDATIONS

All processes in this paper was focused on deriving meaningful and useful information about inside structure of a building in a part. As final product, the list of all walls, ceiling and floor has been extracted. This final product can be extracted in any kind of file formats, which FME software supports, such as Shapefile (SHP) , common CAD formats like DXF/DWG, XLS, Comma Seperated Values (CSV), GML / CityGML. Also list of structural individuals can be also directly imported into any database with spatial data types, such MariaDB, PostgreSQL, Oracle and etc. In this work, it is decided to use CSV file to save attributes of individuals and their geometries, because it has been looked for a way to register structural individuals automatically into Protege software. Also the CSV file format includes a way to store geometries with common GIS standard called Well Known Text (WKT). This human readable standard geometry definition is able to read by most of GIS based software tools and database technologies that supports spatial data. Database technologies are also supporting directly export these spatial data as CSV file format with WKT or Well Known Binary (WKB) formation for geometries.

Imported structural individuals into Protege software are able to represent objects as facet. All of those facets have polygon type geometries and has angular components, which stored as attribute in CSV file format.

With help of reasoners, semantic data can create super classes by evaluating all attributes of facet objects. These super classes are walls, ceiling, floor, room and finally building. Building object has been designed basically as collection of room objects and there is no any suggested automatic tool or algorithm to detect building in the context of the study. Actually, at this point it has been projected that users should add a separate attribute or file name before or after collecting point cloud data. On the other hand, creating a building geometry as union of rooms is also possible, however, it would be not accurate due to some extra spaces and exterior wall thickness.

As a comparison, this study is similar with automatic / semi-automatic CityGML production studies. Also there is another opportunity to adapting this study with CityGML and create rooms as buildingpart class. 3DCityDB, developed by Munich Technical University (TUM), is also able to create buildingparts based on surfaces and store their semantic information to PostgreSQL database (3D City Database for CityGML, 2019) . Semi-automatic methods to derive building information as CityGML format using point cloud and GIS data is already a popular topic and commercial and open source tools like 3dfier supports preferred point cloud processes (Jayaraj and Ramiya, 2018). Although CityGML has a solid ontological structure, it is not possible to use semantic reasoner engines directly in CityGML format.

All of the processes described in this paper was handled in opensource libraries and tools, except FME software. As a suggestion to redesign all progress in open source environment, it is necessary to find an alternative to FME tool. Most probably, pgpointcloud PostgreSQL extension to store point clouds, SFCGAL to create polyhedral surfaces or triangulated surfaces and PostGIS to simplify geometries as planar surfaces with GIS standards will be an ideal toolset and will be compatible with PCL library.

To go a step further in future studies, GeoSPARQL would be ideal query language, which are also able to use simple spatial queries with linked data. Also, Apache Marmotta v3.4 seems as an essential tool to serve linked data by supporting PostgreSQL database and PostGIS functionalities. Apache Marmotta also supports GeoSPARQL after installing kiwi-geosparql module (Apache Marmotta – GeoSPARQL, 2019).

This study has proven that deriving automatic semantic information from point cloud data is possible. In the future, the study will be more developed by introducing relational propositions, topological rules, creating more attributes, such as point intensity, dominant color on surfaces, and using less parameters in decision making algorithms. As a final product, it is created information about building elements. There is the opportunity to store the point cloud data and its relation with automatically created information. This progress is caused to born another idea; point cloud data can be also accepted as most detailed level of LoD concept and semantic information production can be repeated with an iterative method to derive more accurate information.

REFERENCES

3D City Database for CityGML (2019). Retrieved April 2019, from https://www.3dcitydb.org/3dcitydb/fileadmin/downloaddata/3DCityDB_Documentation_v4.2.pdf

Apache Marmotta – GeoSPARQL (2019). Retrieved June 2019, from http://marmotta.apache.org/kiwi/geosparql.html

Cotofrei P., Künzi C., Stoffel K. (2011). Semantic Interpretation of 3D Point Clouds of Historical Objects

Eigen C++ Template Library for Linear Algebra Documentation (2019). Retrieved March 2019, from https://eigen.tuxfamily.org/dox/classEigen_1_1Quaternion.html

Fischler M.A., Bolles R.C. (1981). Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography

Finat J., Delgado F.J. , Martinez R., Hurtado A., Fernandez J.J., San Jose J.I., Martinez J. (2010). Constructors of Geometric Primitives in Domain Ontologies for Urban Environments

Emgard L., Zlatanova S. (2006). Design of an integrated 3D information model.

Cruz C., Marzani F., Boochs F. (2007). Ontology-Driven 3d Reconstruction Of Architectural Objects.

Gallardo Y.P., Crespo A.G., Cuadrado J.L.L., Carrasco I.G. (2015). MESSR: A model-based 3D system for of recognition, semantic annotation and calculating the spatial relationships of a factory's digital facilities

Gurau C., Nüchter A. (2013). Challenges in Using Semantic Knowledge for 3D Object Classification

Jayaraj P., Ramiya A.M. (2018). 3D CityGML Building Modeling from LIDAR Point Cloud Data. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-5

OWL Web Ontology Language Overview (2004). Retrieved June 2019, from https://www.w3.org/TR/2004/REC-owl-features-20040210/

Pangercic, D., Tavcar, R., Tenorth, M., Beetz, M. (2009). Visual scene detection and interpretation using encyclopedic knowledge and formal description logic.

Point Cloud Library Tutorials (2019). Retrieved June 2019, from http://pointclouds.org/documentation/tutorials/

Point Cloud Library 1.8.1 dev documentation (2018). Retrieved April 2018, from http://docs.pointclouds.org/trunk/group__filters.html

Point Cloud Library 1.9.1 dev documentation (2019) Retrieved March 2019, from http://docs.pointclouds.org/trunk/classpcl_1_1_p_c_d_writer.html#a7ce9e829e3830be2be16c12b46ae8a28

Rissanen, J. (1982). A universal prior for integer and estimation by minimum description length. Annals of Statistics, vol. 11, SN. 416–431.

Sanchez V., Zakhor A. (2012) Planar 3D Modeling of Building Interiors from Point Cloud Data

Schnabel R., Wahl R., Klein R. (2008). Efficient RANSAC for Point-Cloud Shape Detection

CURRICULUM VITAE

Name Surname	**: Murat Kendir**
Place and Date of Birth	**: Luebbecke, West Germany / 26.04.1984**
E-Mail	**: muratkendir@gmail.com**

EDUCATION **:**

- **B.Sc.** : 2012, Istanbul Technical University, Faculty of Civil Engineering, Geomatic Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2012-2013 Erg Construction, Deriner Dam, Artvin, Turkey
- 2013 Alarko Contracting Group, Aktogai Copper Mine Project, Kazakhstan
- 2013 – 2017 Sistem Corp. Leica Geosystems Dist., Istanbul, Turkey
- 2017 NetCAD Software Corp., Istanbul, Turkey
- 2018 Ege Real Estate Development Corp., Istanbul, Turkey

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Kendir, M.** 2012. “WEB TABANLI MEKANSAL VERİ YÖNETİM ARACI: MYMAPBASE“