

Report

1. UNSW Lower Campus

1.1 Datasets

Data	Size	#Voxels	Description
dtmbot.xyz	9.22 GB	641,624,355	A terrain with holes, in which the buildings fit
tree.xyz	906 MB	59,640,000	Tree in lower campus
bld1-54.xyz (except for 26 and 46)	3.52GB	241,613,693 in total and 4,646,418 per building	52 buildings in lower campus
be.xyz	249 MB	17,460,029	Built Environment (H13)
blockhouse.xyz	45.4 MB	3,392,202	Blockhouse (G6)
dalton.xyz	25.5 MB	1,887,512	Dalton (F12)
quadrangle.xyz	43.9MB	3,161,733	Quadrangle (E15)
roundhouse.xyz	79.9MB	6,037,174	Roundhouse (E6)
scithe.xyz	17.2MB	1,231,821	Science Theatre (F13)

Note that:

- For GIS-based voxels, its resolution is 20cm. All voxels are recorded in same relative coordinate with offset (336000, 6245250, 20).
- For BIM-based voxels, its resolution is 10 cm. Each building is in its own relative coordinate with MINXYZ.
 - For be.xyz, the offset is (336300, 6245507, 25).
 - For blockhouse.xyz, the offset is (336042, 6245613, 27).
 - For dalton.xyz, the offset is (336305, 6245569, 29).
 - For quadrangle.xyz, the offset is (336409, 6245580, 31).
 - For roundhouse.xyz, the offset is (336047, 6245651, 25).
 - For scithe.xyz, the offset is (336325, 6245582, 28).

1.2 Database Schema

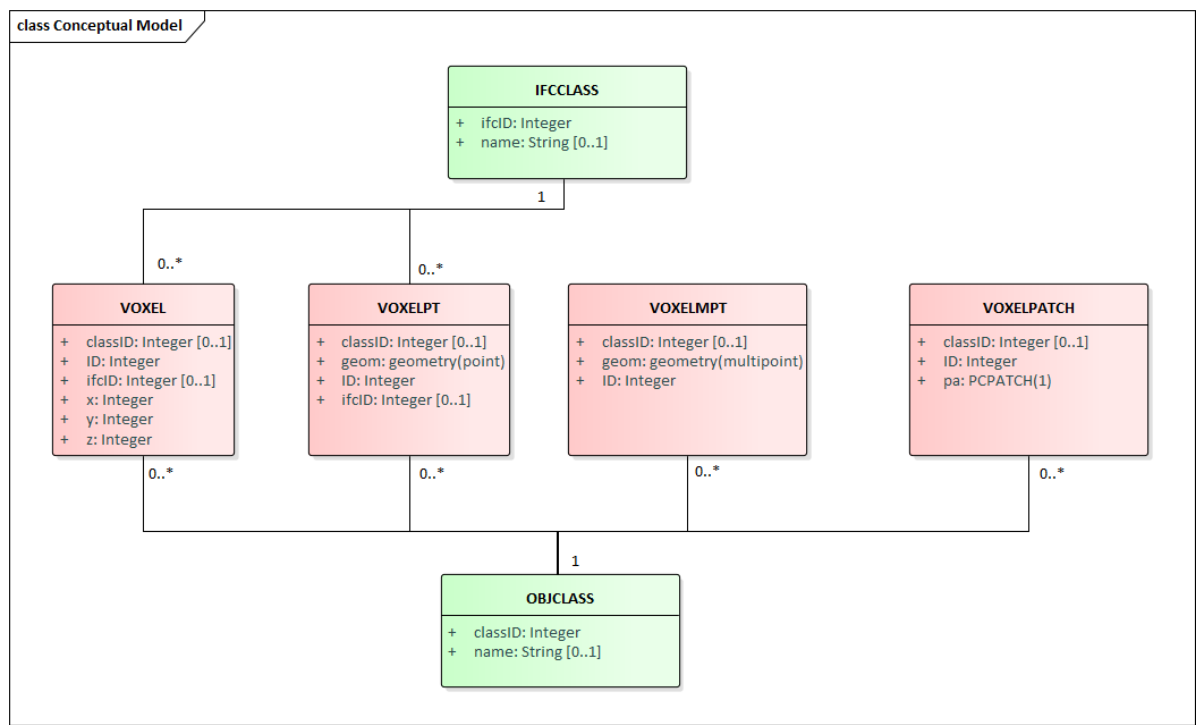


Figure 1. Conceptual Model

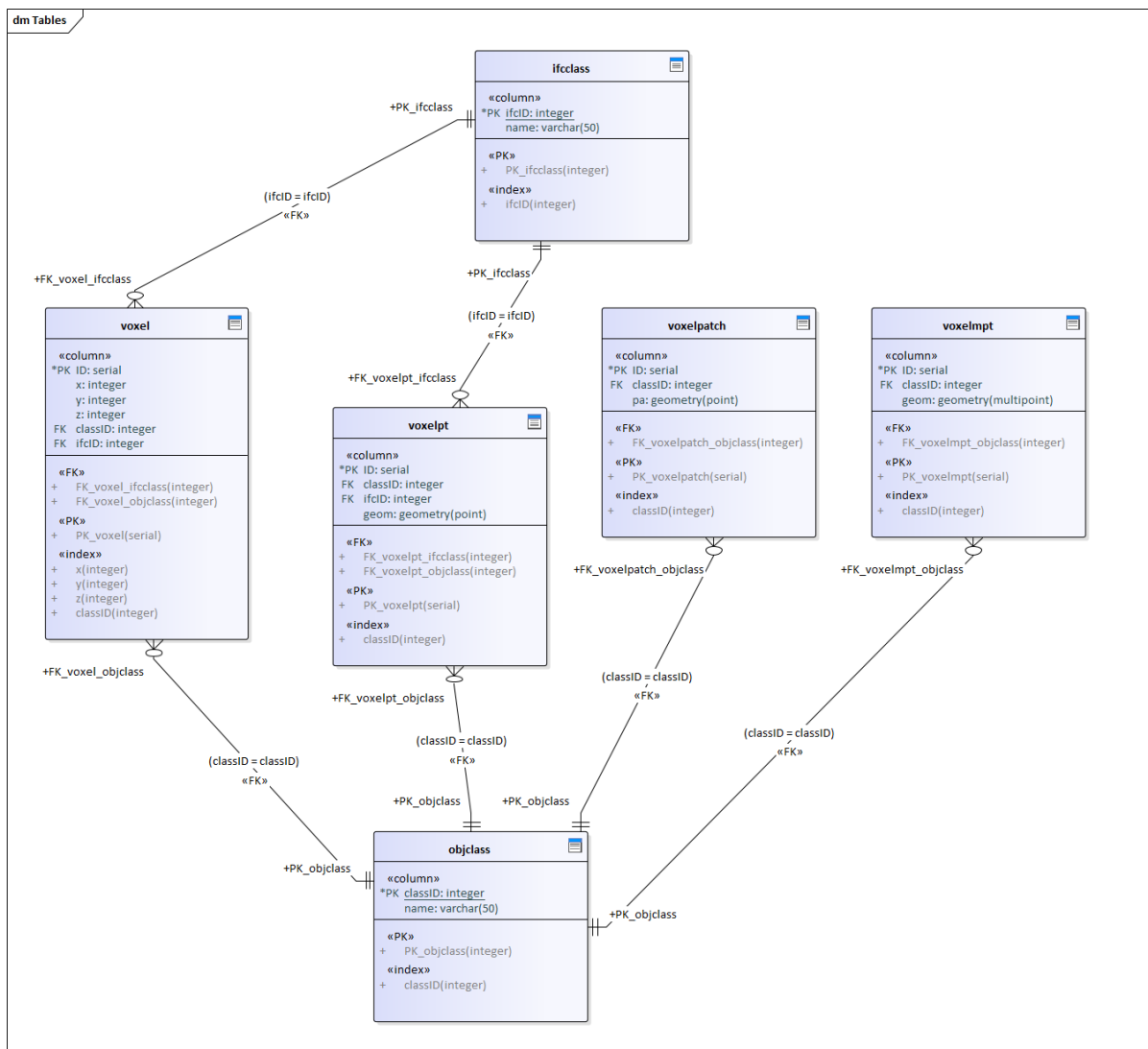








Figure 2. Physical Model

1.3 Data Layout in PostgreSQL

Four Main Tables for voxel storage:

Table	Columns	Geometry	Description				
voxel	id, x, y, z, classid, ifcid	N/A	One voxel per row/record.				
<div><div></div></div>	<div><div>id</div><div>[PK] integer</div></div>	<div><div>x</div><div>integer</div></div>	<div><div>y</div><div>integer</div></div>	<div><div>z</div><div>integer</div></div>	<div><div>classid</div><div>integer</div></div>	<div><div>ifcid</div><div>integer</div></div>	
1	1	1925	391	35	1	[null]	
voxelpt	id, classid, ifcid, geom	POINT(x,y,z)	One voxel per row/record with geometry.				
<div><div></div></div>	<div><div>id</div><div>[PK] integer</div></div>	<div><div>classid</div><div>integer</div></div>	<div><div>ifcid</div><div>integer</div></div>	<div><div>st_astext</div><div>text</div></div>			<div><div></div></div>
1	1	56	[null]	POINT Z (2646 411 84)			
voxelmpt	id, classid, geom	MULTIPOINT	One building per row/record. 20,000,000 point per row/record for “tree” and “dtm”.				
<div><div></div></div>	<div><div>id</div><div>[PK] integer</div></div>	<div><div>classid</div><div>integer</div></div>	<div><div>st_astext</div><div>text</div></div>				<div><div></div></div>
1	55	54	MULTIPOINT Z (1125 2612 31,1125 2612 32,...				
voxelpatch	id, classid, pa	PCPATCH(1)	One building per row/record. 20,000,000 point per row/record for “tree” and “dtm”.				
<div><div></div></div>	<div><div>id</div><div>[PK] integer</div></div>	<div><div>classid</div><div>integer</div></div>	<div><div>pc_astext</div><div>text</div></div>				<div><div></div></div>
1	52	54	{“pcid”:1,“pts”:[[1125,2612,31],[1125,2612,32],[1125,261...				

Two Semantic Tables for IFC and class info:

Table		Columns		Description
ifcclass		ifcid, name		26 IFC class with corresponding name
		ifcid [PK] integer 	name character varying (50) 	
	1	1	IfcBeam	
	2	2	IfcBuildingElementPart	
	3	3	IfcBuildingElementProxy	
objclass		classid, name		classid 1-54 (except for 26 and 46) are building ID. classid 55 is tree ID. classid 56 is dtmbot ID.
		classid [PK] integer 	name character varying (50) 	
	1	55	tree	
	2	56	dtmbot	

2. QGIS Visualization

The data query is processed using a HP laptop. Its processor is Intel(R) Core (TM) i7-7600 CPU @ 2.80GHz and its installed memory is 16.0 GB. Its operating system is 64-bit Windows 10. And the test is performed on PostgreSQL (11.2), PostGIS (2.5.2), and QGIS (3.6.1).

2.1 Sample Visualization

Considering bld52 (we don't have building name at this moment), 29.6MB and 1,881,847 voxels, we extract "geom" first, and then convert its coordinate into EPSG:28356.

```
1. SELECT ST_MakePoint(336000+ST_X(geom)*0.2, 6245250+ST_Y(geom)*0.2, 20+ST_Z(geom)*0.2) AS geom
2. FROM voxelpt
3. WHERE classid=52;
```

Figure 3 shows how to execute query in QGIS.

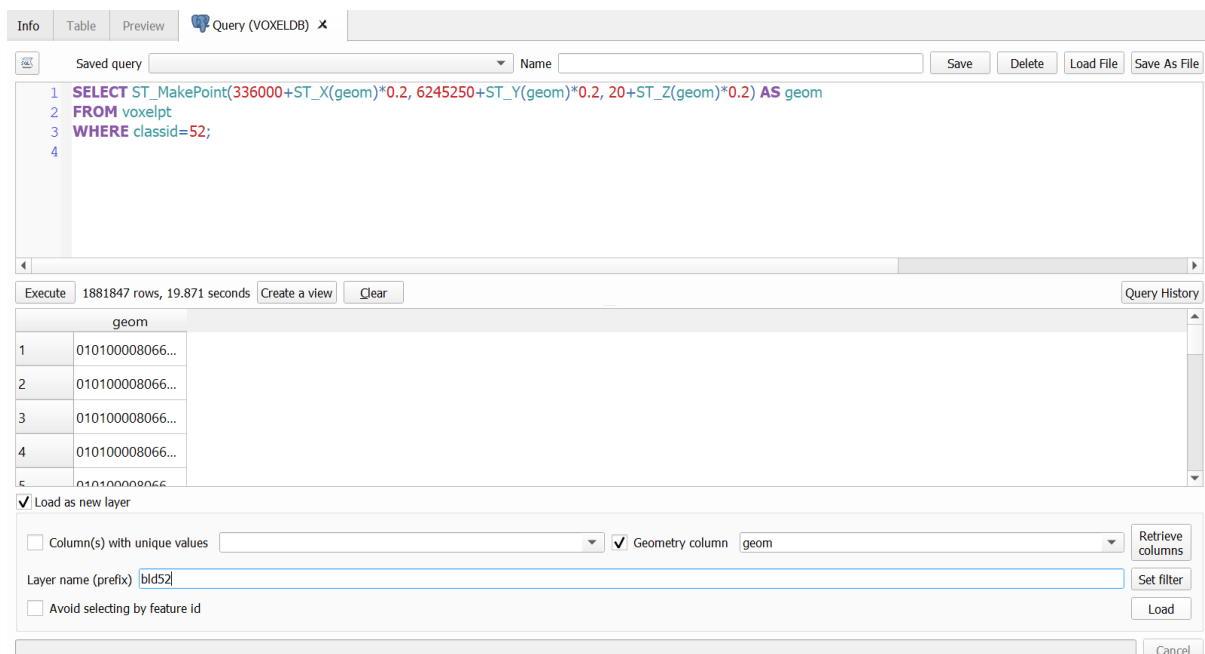


Figure 3. Query execution in QGIS

Note that, loading the above query result as a new layer in QGIS may take several minutes and 3D view is as well. Once choosing 3D view, please keep an eye out for your GPU and memory changes, if you crash, kill the task or stop doing the work at hand and continue to wait patiently. If not necessary or you are not confident in your PC, don't try 3D view.

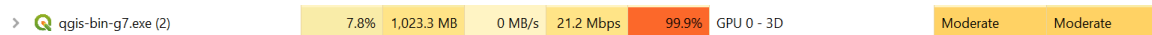




Figure 3. 2D and 3D visualization of “bld52” in QGIS

3. Object Matching between Different Data Source

3.1 Checking Data Info

Data	#Voxels
bld1-54	241,613,693 in total and 4,646,418 per building
tree	59,640,000
dtmbot	641,624,355
Ifcid is not null	33,170,471
be	17,460,029
blockhouse	3,392,202
dalton	1,887,512
quadrangle	3,161,733
roundhouse	6,037,174
scithe	1,231,821

3.2 Assign Temporary classID for IFC buildings

In table “voxel” and “voxelpt”, GIS data occupied 944,110,209 rows.

At this moment, we assume the 6 IFC models with classID 57, 58, 59, 60, 61, and 62, respectively.

```
1. \COPY voxel(x, y, z, ifcID) FROM 'C:\Users\z5039792\Documents\Vox3DMod\data\bim\BE\
   classmodel.xyz' DELIMITER ' ';
2. UPDATE voxel SET classID=57 WHERE classid IS NULL;
3. \COPY voxel(x, y, z, ifcID) FROM 'C:\Users\z5039792\Documents\Vox3DMod\data\bim\Blo
   ckHouse\classmodel.xyz' DELIMITER ' ';
4. UPDATE voxel SET classID=58 WHERE classid IS NULL;
5. \COPY voxel(x, y, z, ifcID) FROM 'C:\Users\z5039792\Documents\Vox3DMod\data\bim\Dal
   ton\classmodel.xyz' DELIMITER ' ';
6. UPDATE voxel SET classID=59 WHERE classid IS NULL;
7. \COPY voxel(x, y, z, ifcID) FROM 'C:\Users\z5039792\Documents\Vox3DMod\data\bim\Qua
   drangle\classmodel.xyz' DELIMITER ' ';
8. UPDATE voxel SET classID=60 WHERE classid IS NULL;
9. \COPY voxel(x, y, z, ifcID) FROM 'C:\Users\z5039792\Documents\Vox3DMod\data\bim\Rou
   ndhouse\classmodel.xyz' DELIMITER ' ';
10. UPDATE voxel SET classID=61 WHERE classid IS NULL;
11. \COPY voxel(x, y, z, ifcID) FROM 'C:\Users\z5039792\Documents\Vox3DMod\data\bim\Sci
   The\classmodel.xyz' DELIMITER ' ';
12. UPDATE voxel SET classID=62 WHERE classid IS NULL;
```



```

COPY 17460029
Time: 406199.727 ms (06:46.200)
UPDATE 17460029
Time: 758352.674 ms (12:38.353)
COPY 3392202
Time: 159865.359 ms (02:39.865)
UPDATE 3392202
Time: 541930.906 ms (09:01.931)
COPY 1887512
Time: 70745.110 ms (01:10.745)
UPDATE 1887512
Time: 441805.998 ms (07:21.806)
COPY 3161733
Time: 269617.281 ms (04:29.617)
UPDATE 3161733
Time: 553708.877 ms (09:13.709)
COPY 6037174
Time: 285602.020 ms (04:45.602)
UPDATE 6037174
Time: 629060.986 ms (10:29.061)
COPY 1231821
Time: 128098.394 ms (02:08.098)
UPDATE 1231821
Time: 529605.717 ms (08:49.606)

```

Figure 4. Log info for IFC data importing

3.3 Update classID for IFC buildings

For BE building, through computing bld19 and its the MAX & MIN (x,y) range in EPSG:28356 CRS, it is easy to find that they are in high probability the same building.

```

1. SELECT MAX(x)*0.1+336300 AS maxx, MIN(x)*0.1+336300 AS minx, MAX(y)*0.1+6245507 AS
   maxy, MIN(y)*0.1+6245507 AS miny
2. FROM voxel
3. WHERE classid=57;
4.
5. SELECT MAX(x)*0.2+336000 AS maxx, MIN(x)*0.2+336000 AS minx, MAX(y)*0.2+6245250 AS
   maxy, MIN(y)*0.2+6245250 AS miny
6. FROM voxel
7. WHERE classID=19;

```

	maxx numeric	minx numeric	maxy numeric	miny numeric
1	336450.8	336301.6	6245552.9	6245508.6
1	336384.8	336300.8	6245552.4	6245519.4

Figure 5. (x,y) range for BE in EPSG:28356 CRS

Then, visualizing above two buildings in CloudCompare, it looks similar, at least in shape.

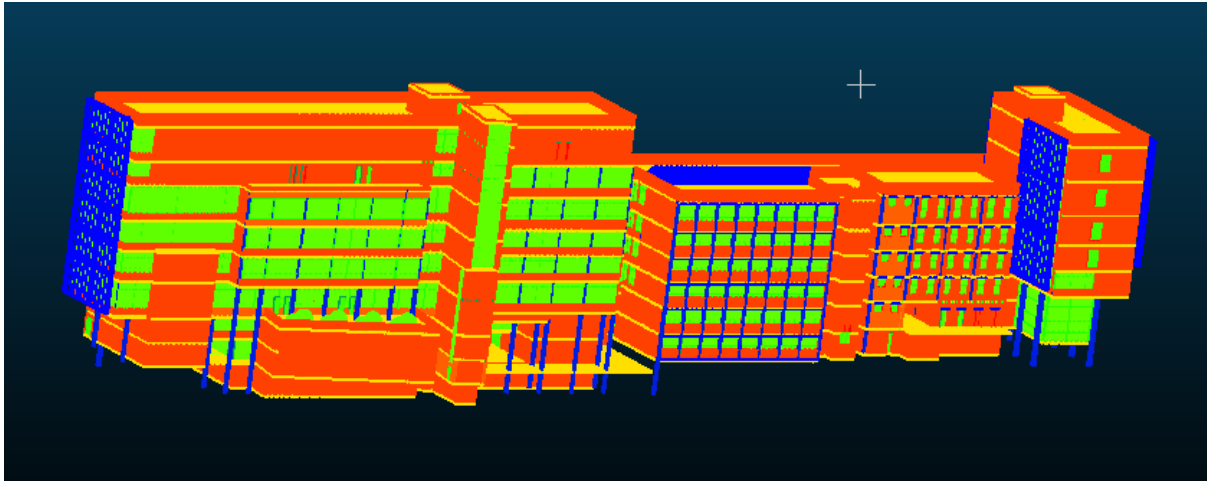


Figure 6. BE building in CloudCompare

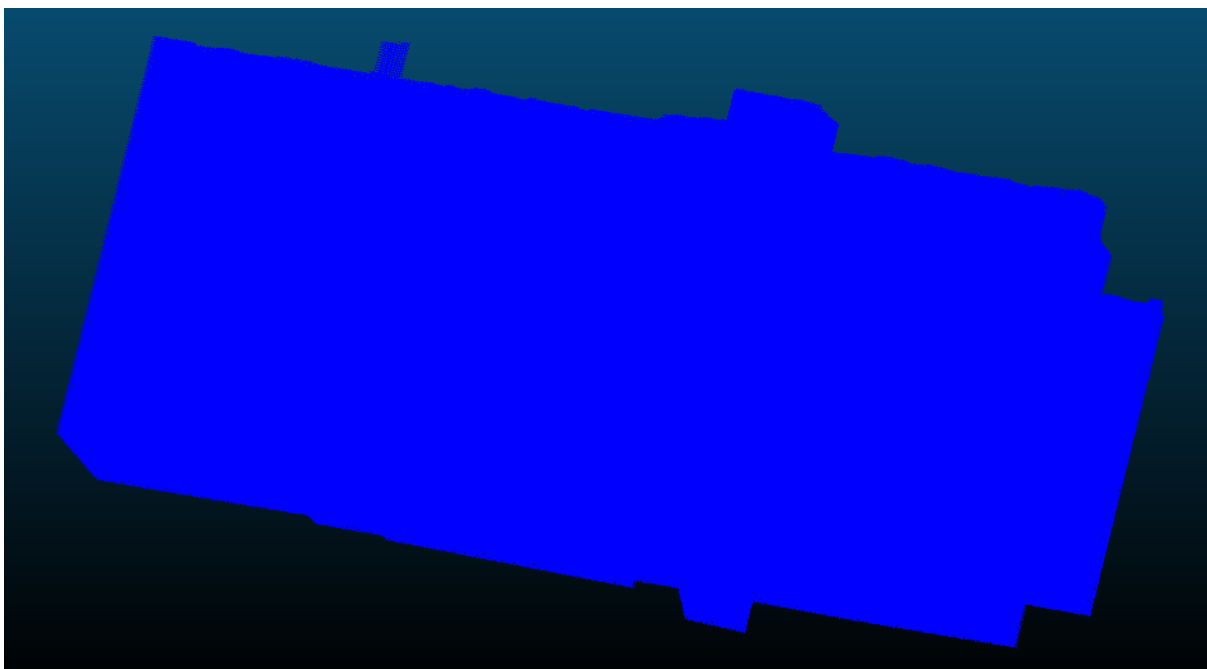


Figure 7. bld19 in CloudCompare

For Blockhouse, Dalton, Quadrangle, Roundhouse, SciThe buildings, calculating (x,y) range for all buildings with classID<=54. And then retrieve same range for above 5 buildings to do matching.

```

1. SELECT MAX(x)*0.1+336042 AS maxx, MIN(x)*0.1+336042 AS minx, MAX(y)*0.1+6245613 AS
   maxy, MIN(y)*0.1+6245613 AS miny
2. FROM voxel
3. WHERE classid=58;
4.
5. SELECT MAX(x)*0.1+336305 AS maxx, MIN(x)*0.1+336305 AS minx, MAX(y)*0.1+6245569 AS
   maxy, MIN(y)*0.1+6245569 AS miny
6. FROM voxel
7. WHERE classid=59;
8.
9. SELECT MAX(x)*0.1+336409 AS maxx, MIN(x)*0.1+336409 AS minx, MAX(y)*0.1+6245580 AS
   maxy, MIN(y)*0.1+6245580 AS miny
10. FROM voxel
11. WHERE classid=60;
12.

```

```

13. SELECT MAX(x)*0.1+336047 AS maxx, MIN(x)*0.1+336047 AS minx, MAX(y)*0.1+6245651 AS
    maxy, MIN(y)*0.1+6245651 AS miny
14. FROM voxel
15. WHERE classid=61;
16.
17. SELECT MAX(x)*0.1+336325 AS maxx, MIN(x)*0.1+336325 AS minx, MAX(y)*0.1+6245582 AS
    maxy, MIN(y)*0.1+6245582 AS miny
18. FROM voxel
19. WHERE classid=62;
20.
21. SELECT MAX(x)*0.2+336000 AS maxx, MIN(x)*0.2+336000 AS minx, MAX(y)*0.2+6245250 AS
    maxy, MIN(y)*0.2+6245250 AS miny
22. FROM voxel
23. WHERE classID<=54
24. GROUP BY classID;

```

Building Name	Range in (x,y) EPSG:28356 CRS				
Blockhouse		maxx numeric	minx numeric	maxy numeric	miny numeric
	1	336126.3	336043.0	6245650.8	6245614.9
Dalton		maxx numeric	minx numeric	maxy numeric	miny numeric
	1	336332.4	336306.0	6245643.1	6245570.7
Quadrangle		maxx numeric	minx numeric	maxy numeric	miny numeric
	1	336501.5	336410.1	6245626.4	6245581.7
Roundhouse		maxx numeric	minx numeric	maxy numeric	miny numeric
	1	336125.8	336048.0	6245749.7	6245652.9
SciThe		maxx numeric	minx numeric	maxy numeric	miny numeric
	1	336380.3	336326.9	6245633.4	6245583.5

	maxx numeric	minx numeric	maxy numeric	miny numeric
1	336447.4	336385.0	6245404.8	6245321.4
2	336135.8	336085.8	6245789.2	6245746.2
3	336483.0	336222.0	6245811.4	6245730.0
4	336525.6	336450.0	6245403.8	6245313.4
5	336207.4	336125.8	6245668.0	6245607.2
6	336121.4	336045.4	6245647.8	6245623.2
7	336156.8	336122.0	6245573.2	6245534.8
8	336380.0	336338.8	6245634.0	6245597.2
9	336331.6	336308.2	6245646.6	6245575.4
10	336516.6	336462.6	6245500.2	6245411.6
11	336555.4	336506.8	6245495.6	6245383.4
12	336397.4	336285.2	6245684.4	6245643.2
13	336538.8	336394.4	6245675.4	6245573.6
14	336572.2	336480.4	6245639.2	6245541.0
15	336519.4	336509.4	6245578.0	6245568.6
16	336473.0	336368.4	6245592.4	6245559.2
17	336449.6	336429.2	6245605.2	6245582.2
18	336449.8	336382.6	6245545.4	6245510.0
19	336384.8	336300.8	6245552.4	6245519.4
20	336314.8	336293.6	6245539.2	6245474.2
21	336403.6	336376.6	6245519.2	6245499.8
22	336404.6	336374.8	6245501.2	6245470.6
23	336458.4	336307.6	6245522.8	6245441.4

24	336190.6	336134.6	6245698.0	6245669.8
25	336309.4	336219.6	6245641.8	6245589.8
26	336205.2	336146.6	6245822.8	6245754.6
27	336106.6	336062.2	6245439.8	6245374.0
28	336093.8	336048.8	6245513.4	6245444.4
29	336225.6	336146.4	6245397.6	6245353.4
30	336348.4	336227.4	6245383.6	6245340.6
31	336366.8	336352.2	6245366.6	6245358.2
32	336364.8	336357.0	6245353.6	6245345.6
33	336148.8	336112.4	6245398.8	6245364.2
34	336169.4	336144.0	6245740.4	6245705.6
35	336049.2	336000.0	6245846.2	6245800.0
36	336138.8	336051.8	6245839.8	6245794.4
37	336119.8	336032.4	6245585.4	6245511.4
38	336517.2	336478.0	6245706.4	6245667.0
39	336228.4	336199.6	6245746.2	6245709.6
40	336460.6	336359.0	6245730.4	6245677.2
41	336450.8	336426.6	6245776.0	6245754.6
42	336448.0	336432.2	6245729.4	6245713.0
43	336130.4	336051.8	6245740.6	6245660.2
44	336040.2	336009.8	6245735.2	6245698.6
45	336529.6	336503.0	6245600.2	6245577.0
46	336527.4	336489.6	6245780.4	6245726.8
47	336563.8	336529.4	6245774.8	6245720.2
48	336594.4	336562.0	6245770.6	6245713.6
49	336586.2	336553.2	6245709.6	6245653.4
50	336554.6	336521.6	6245716.8	6245662.2
51	336481.0	336458.6	6245771.8	6245754.6
52	336236.6	336225.0	6245779.6	6245771.0

Figure 8. (x,y) range for all 52 building

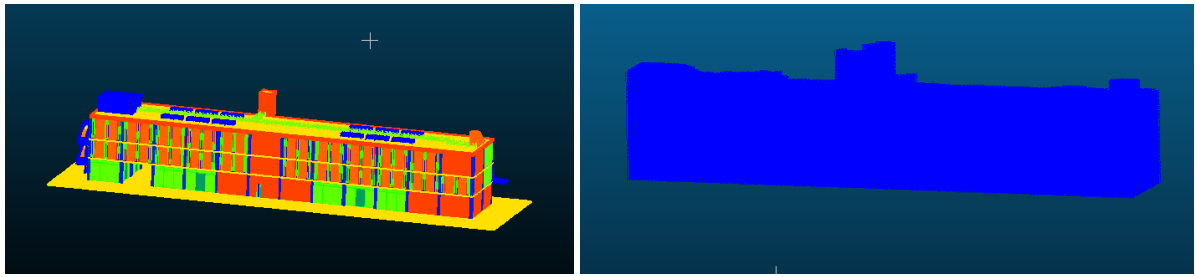


Figure 9. Blockhouse

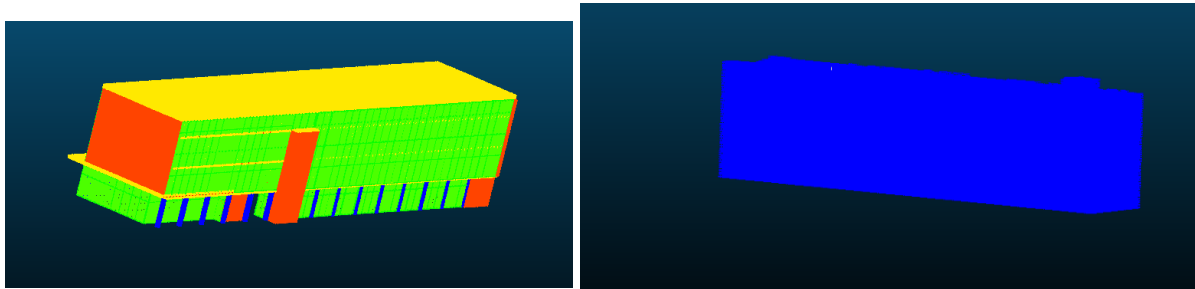


Figure 10. Dalton

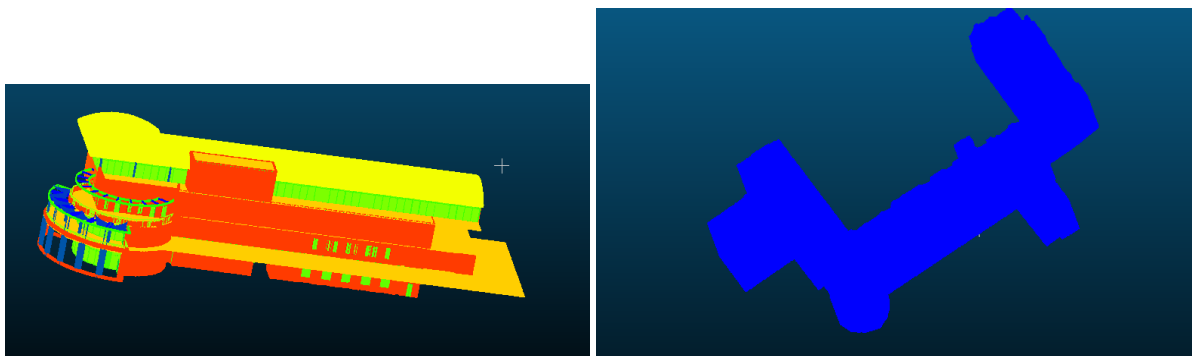


Figure 11. Quadrangle

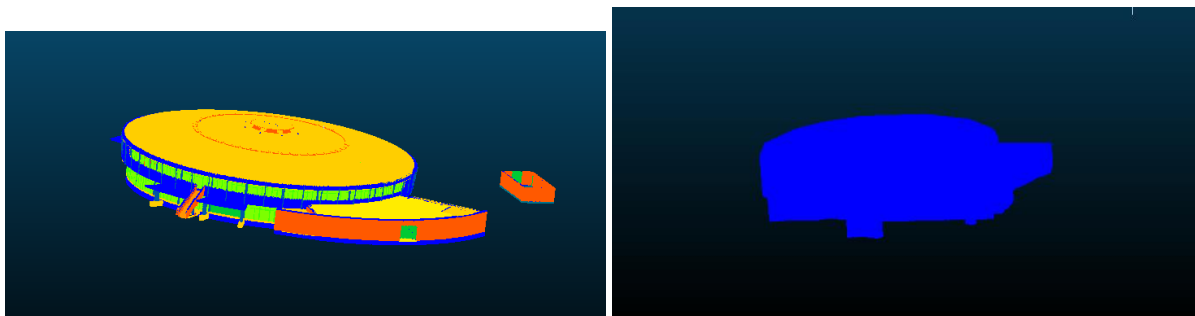


Figure 12. Roundhouse

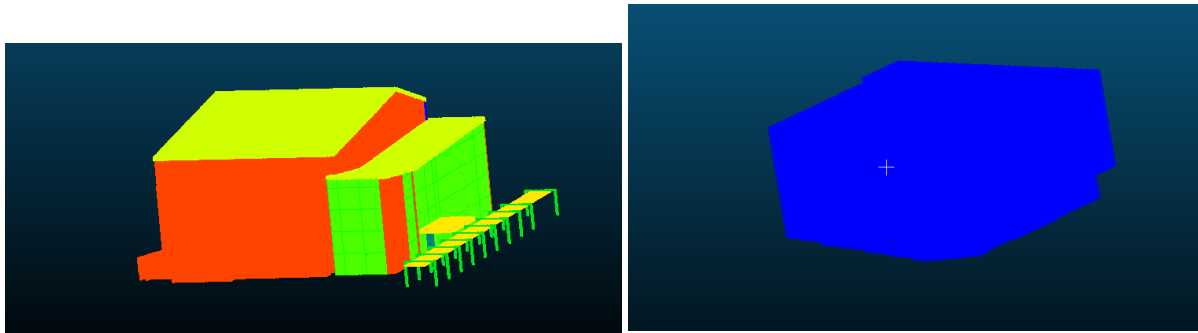


Figure 13. Science Theatre

In summary, the corresponding bld are list in below table:

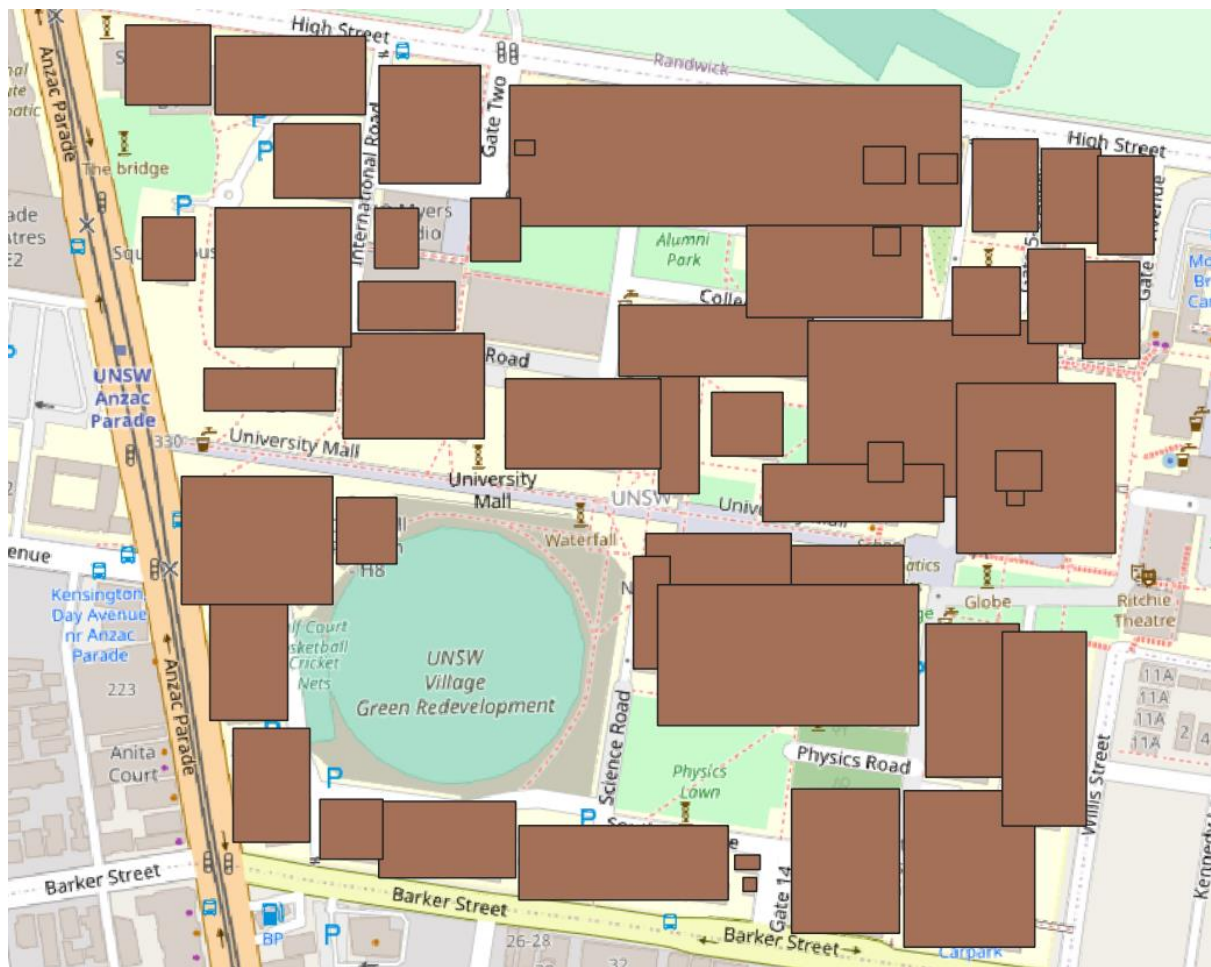
Id	Name	Num	Old classID	New classID
Bld19	Built Environment	H13	57	19
Bld6	Blockhouse	G6	58	6
Bld9	Dalton	F12	59	9
Bld13	Quadrangle	E15	60	13
Bld44	Roundhouse	E6	61	44
Bld8	Science Theatre	F13	62	8

Update the 6 buildings:

1.	UPDATE	voxel	SET	classID=19	WHERE	classid=57;
2.	UPDATE	voxel	SET	classID=6	WHERE	classid=58;
3.	UPDATE	voxel	SET	classID=9	WHERE	classid=59;
4.	UPDATE	voxel	SET	classID=13	WHERE	classid=60;
5.	UPDATE	voxel	SET	classID=44	WHERE	classid=61;
6.	UPDATE	voxel	SET	classID=8	WHERE	classid=62;
Time: 1h 12m 19s						

3.4 Assign Name for Each Building in Lower Campus

Ignore ...





4. Conversion to Point

In this section, we consider the second data layout, that stores each voxel as a geometry POINT including (x,y,z).

```
1. DELETE FROM voxelpt;  
2. INSERT INTO voxelpt(classID, ifcID, geom) SELECT classID, ifcID, ST_MakePoint(x,y,z) FROM voxel AS VALUES;
```

Time: 3h 18m 12 s

5. Conversion to Multipoint

To enable the use of geometrical functions from PostGIS, POINTs are transformed into MULTIPOINT type, which is a collection of POINTs.

5.1 Create table

```
1. DROP TABLE IF EXISTS voxelmp CASCADE;
2. CREATE TABLE voxelmp
3. (
4.     id serial PRIMARY KEY,
5.     classID INTEGER,
6.     ifcID INTEGER,
7.     geom geometry
8. );
```

5.2 Partition Principles

Rule-1: For general building objects without IFC features, each building is one Multipoint.

Rule-2: For building objects with IFC objects, we regard each IFC object as one Multipoint.

Rule-3: For tree and dtmbot, we combine GIS dataset to decide the patch size.

5.3 Data Generation

First, for general building voxels without IFC semantic information, we straightforward collect all POINTs in “voxelpt” with same classID into one MULTIPOINT geometry.

```
1. DO $$
2. BEGIN
3.     FOR idx in 1..54
4.     LOOP
5.         IF idx = 26 OR idx = 46 THEN
6.             raise notice 'The buidling % could not be found', idx;
7.         ELSE
8.             INSERT INTO voxelmp(classID, geom)
9.             VALUES (idx, ST_Collect(ARRAY(SELECT geom FROM voxelpt WHERE classID=idx
10.             AND ifcID IS NULL)));
11.         END IF;
12.     END LOOP;
13. END;
13. $$
```

Time: 14h 47m 56s

Next, for building objects with IFC features, each partition is a collection of IFC voxels with same ifcID.

```
1. DO $$
2. DECLARE
3.     f record;
4. BEGIN
5.     FOR f in SELECT DISTINCT classID, ifcID
6.             FROM voxel
7.             WHERE ifcID IS NOT NULL
8.     LOOP
9.         INSERT INTO testmpt(classID, ifcID, geom)
```

```

10.         VALUES (f.classID, f.ifcID, ST_Collect(ARRAY(SELECT geom FROM voxelpt WHE
        RE classID=f.classID AND ifcID=f.ifcID)));
11.     END LOOP;
12. END;
13. $$

```

Time: 97 min 19 secs.

Last, for tree, there is another tree data set. It consists of points, which represent the trunk of the tree. So using this point as a center and assuming an horizontal radius (3-4m) we can try to partition the trees.

Actually, we have to create terrain objects with respect to the surface objects as paths, gardens, roads. Jinjin has these vector non-overlapping polygons. You can use them to find ‘all voxels of the dtm in a specific polygon’ and assign the semantic of the polygon. Then the dtm will be not partitioned randomly but according to the surface objects.

```

1. pg_dump -U postgres -h 149.171.16.253 -p 5432 -t tree crc_lcl_proj | psql -
  h 149.171.16.253 -p 5433 -U postgres -d voxeldb
2. pg_dump -U postgres -h 149.171.16.253 -p 5432 -t terrain crc_lcl_proj | psql -
  h 149.171.16.253 -p 5433 -U postgres -d voxeldb

```

For tree, we test the first 10 points and visualize them in QGIS.

```

1. SELECT ST_MakePoint(336000+ST_X(geom)*0.2, 6245250+ST_Y(geom)*0.2, 20+ST_Z(geom)*0.
  2) AS geom
2. FROM voxelpt
3. WHERE classid=55
4. LIMIT 5;
5. SELECT geom FROM tree;

```

```

SELECT ST_MakePoint(336000+ST_X(geom)*0.2, 6245250+ST_Y(geom)*0.2,
20+ST_Z(geom)*0.2) AS geom

```

```

FROM voxelpt

```

```

WHERE classid=55

```

```

LIMIT 5;

```

```

SELECT geom FROM tree;

```

6. Conversion to PCPATCH

PostgreSQL Pointcloud deals with all this variability by using a "schema document" to describe the contents of any particular point. Each point contains a number of dimensions, and each dimension can be of any data type, **with scaling and/or offsets** applied to move between the actual value and the value stored in the database. The schema document format used by PostgreSQL Pointcloud is the same one used by the PDAL library.

Schema documents are stored in the `pointcloud_formats` table, along with a `pcid` or "pointcloud identifier". Rather than store the whole schema information with each database object, each object just has a `pcid`, which serves as a key to find the schema in `pointcloud_formats`. This is similar to the way the `srid` is resolved for spatial reference system support in PostGIS.

The central role of the schema document in interpreting the contents of a point cloud object means that care must be taken to ensure that the right `pcid` reference is being used in objects, and that it references a valid schema document in the `pointcloud_formats` table.