Technical University of Munich

School of Engineering and Design

Chair of Computational Modeling and Simulation

Software Lab 2021, Group 8

Model-Based Scan Planning

Student:
Mengying Jia
mengying.jia@tum.de

Supervised by:
Florian Noichl. M.Sc.
Yuandong Pan, M.Sc

January, 2022

Contents

1 Introduction

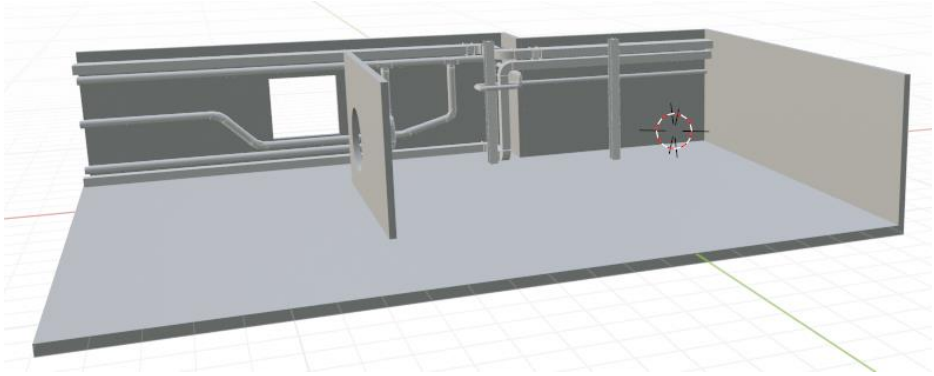
1.1 Background of scan planning

In Architecture, Engineering and Construction field the sensor technology has played an important role in recent decades. With the acquisition of 3D data in an accurate and quick way the process of design and construction of architectures has been improved. However usually scanning the constructions is a time-consuming work, in order to reduce the scanning time and costs and meanwhile acquire as accurate information of the structures as possible is a nowadays a highly discussed topic[1].

The aim of *Model-based Scan Planning* is to get an optimized scanning plan, which has a balance between computation cost for the scanning plan and accuracy of the scanning results, by simulating the scanning progress using the BIM (Building Information Models) method.

1.2 Target of the project

In this project the main purpose is to find an optimized scanning plan for a single room, which means finding a minimized number of scanning points on the floor of the room to acquire as much and accurate information as possible of the room. The 3D model of the room shows in figure 1[Φ].



Length = 21.6176m, Width = 14.9463m, Height = 4.4632m

Fig. 1.1 Target 3D BIM model of a room

Usually there are some requirements for the scanning data acquired through the optimized scanning plan and some parameters to evaluate performances of the scanning plan, as shown below[2]:

Requirements for scanning data quality:

- **LOA** (Level of Accuracy): Tolerance of positioning accuracy of each individual point in 3D point cloud data. LOA is typically defined in millimeter
- **LOD** (Level of Density): Minimum object size that can be extracted from the point clouds. LOD relates to surface sampling, i. e. how dense the scanned points are. LOD is thus typically defined as a distance (in millimeter) between neighboring scanned points

Parameters to evaluate scanning plan:

- **Number of scanning points**: usually a smallest number of scanning points is preferred in order to reduce cost and simplify the scanning plan
- **Acquired rate**: Percent of the covered geometric data of the scanning plan in the total model, which presents the degree of completeness for the scanning plan
- **Data-overlap status**: the overlap status expresses by how many times one element in the model will be scanned to present the degree of accuracy of the scanning plan
- **Computation time**: Time that the optimization algorithm uses to get the scanning plan

[Φ] Via this link in order to view more information of the target model:
https://view.tridify.com/bim/#/conversion/M8fgGNu3TPkT0-ingY_Dj8yto2KeLjcCtGfH4BfRTws

In this project we choose GSA(US General Services Administration) Level 1 as the LOA and LOD standards for the scan data quality, which shows in Table 1 and set the acquired to 90% as the boundary condition for the optimization algorithm when minimizing the number of scanning positions.

Table 1.1 Data quality requirements standardized by GSA[2]

GSA Level	LOA (Tolerance) /mm	LOD (Data Density) /mm × mm
1	± 51	152×152

1.3 Reference device parameter

In order to set parameters in the scan planning progress and check the feasibility of the , I choose a type of modern laser scanner, Leica ScanStation P30/P40 as reference. This scanner could cover a point of view 360◦ horizontally and around 290◦ vertically of ranges of up to 270m with a 3D data accuracy of $1.2\text{mm} \pm 10\text{ppm}$, and in the 50-meter view the accuracy could reach 0.5mm.

Fig. 1.2. 3D laser scanner Leica ScanStation P30/P40

1.4 Software used

In this project I use software Blender 2.93.1 to import the BIM model of the target room and the blender python API to get the scanning data for each candidate scanning position. Then import the scanning data into software Matlab 2021b to implement the optimization algorithm and evaluate the results.

2 Workflow of the project

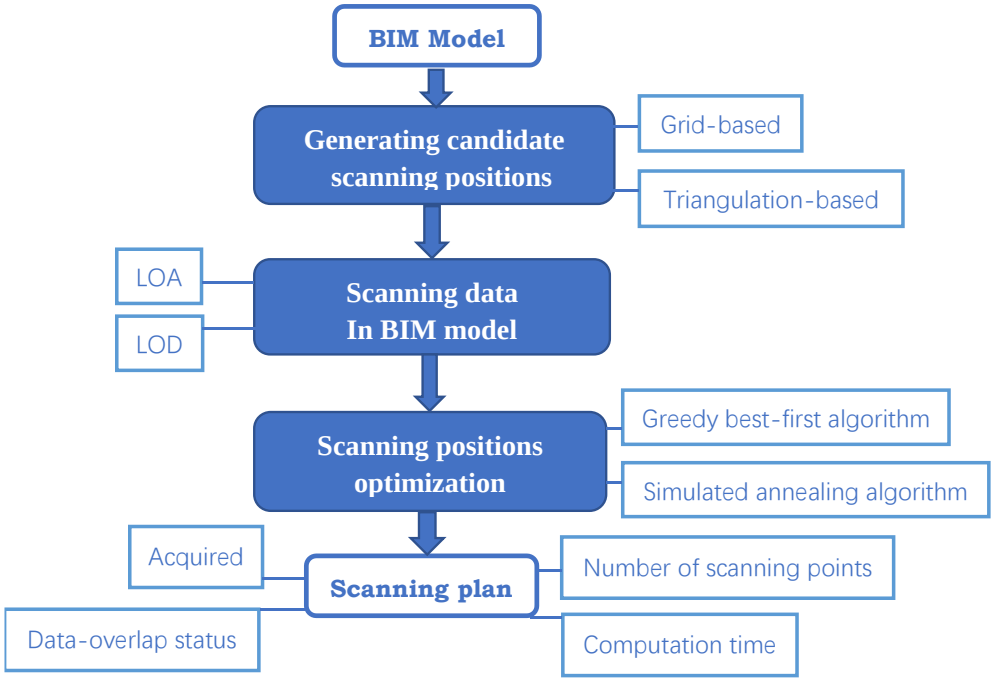


Fig. 2.1 Workflow

2.1 Generating candidate scan positions

After deciding the target structure model and import it into a BIM software, a relatively large number of candidate scanning positions will be generated for further optimization of the scanning plan. Usually, grid-based method, triangulation-based method or a combination of two methods are used to generate candidate points[3]. In this project I choose grid-based method on a floor plan to create 28 candidate scanning points, as shown in Fig. 2.2. The x and y components in Blender are defined as below (assume $z \equiv 0$, in this case there is no standing support for the scanner and the height of the scanner below the scanning origin is neglectable),

Coordinates for candidate scanning points:
cam_position_0=np.array([-5,-2,1,5,8,11,14]), cam_position_1=np.array([-1,2,5,8])

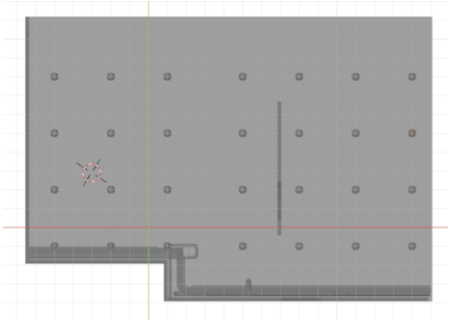


Fig. 2.2 Generating candidate scanning positions with grid-based method

2.2 Scanning information simulation

In order to get an optimized scanning plan, the information that each candidate scanning point could cover are needed. So the next step is to get scanning data for each candidate position in BIM software. In this project I choose vertices as unite element to represent the scanning information for a certain candidate scanning point, and store the visibility for vertices to each candidate point in a data set called **PVS (Potential Visible Set)** as input of the optimization algorithm for the scanning plan[4]. The data size of a PVS is number of vertices×number of candidate scanning points.

2.3 Minimizing scanning points

With all the candidate scanning positions and the data of the model they could cover, I compare the data for one point with each other and the whole target model using optimization algorithms, for example, simulated annealing algorithm and greedy best-first algorithm, to get a minimized number of scanning positions as the result scanning plan at a required degree of data completeness for the original model. The mathematical expression for the aim of optimization algorithm shows as below[5]:

P is the set of all candidate scanning positions, which contains n points denoted as $p_i(x_i, y_i)$

$$p_i(x_i, y_i) = \begin{cases} 0 & if\ \ p_i \in S \\ 1 & otherwise \end{cases}$$

S is the set of selected scanning positions applied for the optimization algorithm
The aim of the optimization algorithm is to

$$minimize \sum_{i}^{n} p_i(x_i, y_i)$$

meanwhile

$$keep \quad \theta \geq \theta_{req}$$

- *θ(Acquired rate of the scanning plan): number of covered vertices/number of all vertices in target model*
- *θ_{std}: required acquired rate of the scanning plan*

2.4 Evaluating results

To evaluate the scanning plan's quality, some parameters are considered. The **number of scanning points** of the plan and **computation time** of the optimization progress show the workload of the scanning plan. The **acquired rate**(%) and **data-overlap status**(average scanned times for vertices covered in a scanning plan) to illustrate the data completeness of the scanning plan.

3 Scanning data access

To illustrate the geometry of the solid model for the target room, Blender uses a mesh including *82,281* polygons, *126,915* edges and *44,781* vertices to present the surfaces of the model, as we can see in Fig. 3.1 and set the maximal length of an edge to *0.1m*, which meets the *152mm* requirement for **LOD** (seen in Table 1.1). The aim is to check the vertices' visibility to each candidate points and get the **PVS** with the data size of *28×44,781* for the further optimization algorithm.

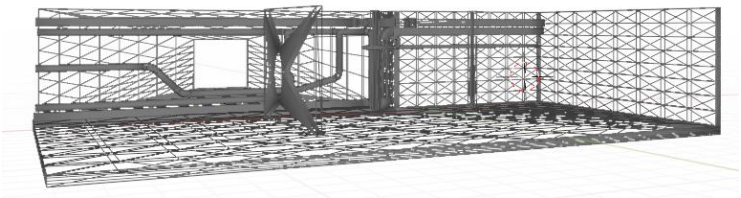


Fig. 3.1 Mesh mode of the target room in BIM

Generally, a point in the scene is considered visible or 'scannable' if it is within scanning distance and without occlusion from the selected scanning location. Usually, the following three criteria are used to check a vertice's visibility[2]:

- ***Depth of Field (DOF)****: Only points within the minimum and maximum scanning distances of the scanner can be acquired. DOF varies for different types of scanners. Since the maximal scanning distance of the reference scanner Leica ScanStation P30/P40 is 270m, which is much larger than the magnitudes of the room, the DOF would not be considered during the visibility check process.*
- ***Field of View (FOV)****: Only points within the vertical and horizontal angle ranges of the scanner can be acquired. These ranges result from each scanner's physical and mechanical characteristics. The reference laser scanners. Leica ScanStation P30/P40 can cover 360◦ horizontally and around 290◦ vertically, so during the visibility check process I assume all vertices of the model are in the FOV of the scanner.*
- ***Line of Sight****: Only points with direct line of sight from the scanning location can be acquired. In this project the* ***ray_cast*** *function from the BVHTree class in Blender Python API's bpy module will be used to check the Line of Sight for the vertice's visibility check.*

As discussed before, the ray_cast function from BVHTree class in bpy module is used to check whether a vertice is visible or scannable to a scanning point, the usage of this function illustrates as below,

#in order to use the ray_cast function, an object bvh need to be defined first,
bvh = BVHTree.FromPolygons(vertsInWorld, [p.vertices for p in obj.data.polygons]);
amongs "vertsInWorld" is a list stored position information for all 44,781 vertices
obj.data.polgons is the call for all polygons of the target object "room"
#"bvh" is an object stores relationship of polygons using data structure BVH Tree

#A is a vertice and its index is num and use the ray_cast function to check the visibility of A from a certain scanning location,

```
A = vertices[num]
location_A, normal_A, index_A, distance_A
= bvh.ray_cast(cam.location,(A-cam.location).normalized());
#the input variables are the origin position of the light, i.e. the position of a certain scanning point, and the direction of the checking ray;
#the return values are the central location, normal vector, index and distance from origin to center of the first polygon the ray hits;
all return values will be none if there is no hit.
#the vertice is visible when there's no hit:
if not location_A:
    visibility[num] = True;
#If the vertice is close to the polygon that the ray hits first,
the vertice is assumed to be the hit and still visible
limit=0.05
elif (A - location_A).length <= limit:
    visibility[num] = True;
```

The limit is defined as 0.05m, which meets the requirement of **LOA** for ±51mm (seen in Table 1.1). And the 3D data accuracy of the reference laser scanner could reach 0.5mm in the 50-meter view (seen in chapter 1.3), which could easily meet this requirement.

The limit check is necessary, because the number of vertices of a scanning point could cover is extremely low without the limit check, due to the structure of BVHTree defined in bpy; and the results applying the limit check consistent with common sense. This choice is kind of conservative because the maximal length of an edge is 0.1m and there is a possibility that a visible vertice is set to unvisible.

4 Optimization algorithm for scan planning

4.1 Simulated annealing algorithm

The thought of simulated annealing algorithm is to discard one candidate that makes least contribution to the acquired rate over one loop, until the acquired rate could not reach the require value. The thoughts of this optimization algorithm shows below[5].

```
do
{
    for p_i(x_i, y_i) in P,
        calculate acquired rate with out p_i(x_i, y_i)
    discard p_i(x_i, y_i) with the highest value of acquired rate
    re-define candidate set P(n points → n − 1 points)
    candidates, without which acquired rate couldnot reach requirement are set into solution set S
}
while acquired rate(S) >= required acquired rate
```

4.2 Greedy best-first algorithm

The thought of greedy best-first algorithm is first find the scanning that covers the highest number of vertices of the object, and then add other candidates one by one due to the increment to the acquired rate, until the acquired rate could reach the purpose value, the thoughts of this optimization algorithm shows below[5].

```
for i = 1:.number_of_candidates-1
    %loop ends when reaching the requirement
    if current acquired rate reaches required value
        break;
    for j = 1:number_of_candidates-i
        %check which of the rest points could increase the most indices to the solution
        increment(j)
    candidate[best] has the maximal increment
    set candidate[best] into the solution set
    delete the data of points already in the solution from the current pvs for next loop
```

It's obvious that the simulated annealing algorithm needs more calculation than the greedy best-first algorithm, so I choose to use the algorithm for "adding" instead of "removing".

4.3 Results

For the results and evaluations of a scanning plan I focus mainly on the following **parameters**:

1) **number of scanning points**: the minimized number of scanning positions that could reach required acquired
2) **acquired rate**: the rate of covered vertices in the scanning plan
3) **data-overlap status**: the average value of how many times a vertice has been scanned if it 's covered in the scanning plan
4) **computation time**: use cputime to calculate the computation time of the optimization progress

The **running results** of greedy best-first algorithm in matlab 2021b:

1) Required acquired = 0.9

First I set the required acquired rate to 0.9, the results are shown in figure 4.1 and it could not get the wanted scanning plans with the current data.

```
struct for "scan" built
data for the struct set
the computation time of the optimization algorithm is 327.593750
the acquired rate of this plan is 0.605949
which could not meet the value of required acquired rate
this method has failed with the current candidate points and optimization algorithm
greedy best-first algorithm completed
```

Fig. 4.1 greedy best-first, required acquired rate=0.9

To analyze this situation, I have found that the order of magnitude for the indices for curved surfaces of the pipes are much larger than other surfaces, as shown in figure 4.2. Meanwhile the back side of the pipes are on the wall (seen in figure 1.1) and could not be scanned, so the acquired rate has been a relatively lower number.

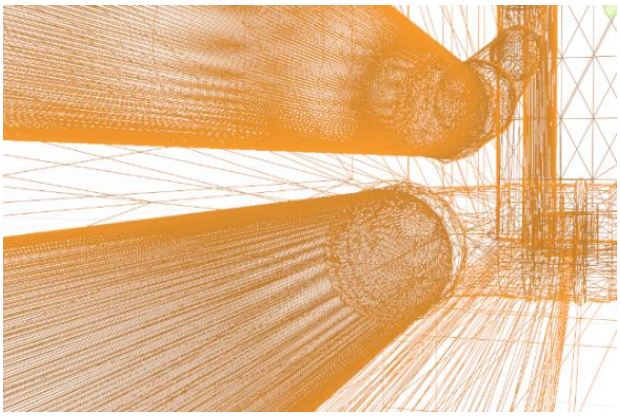


Fig. 4.2 mesh for pipes

2) Required acquired = 0.55

Then I set the required acquired rate to 0.55 and got the results shown in figure 4.3

```
the number of scanning points of this plan is 10
positions for scanning points in the solution set are
x:    -5    -2    -2     1     8    11    11    14    14    14

y:     2     2     5     8    -1    -1     2    -1     2     5

the acquired rate of this plan is 0.555749
the data overlap status of this plan is 2.677502
the computation time of the optimization algorithm is 162.531250
greedy best-first algorithm completed
```

Fig. 4.3 greedy best-first, required acquired rate=0.55

To check whether this result is valid I import the scannable vertices into blender and show the covered vertices, as shown in figure 4.4, in which the mesh is still relatively complete. And the scanning positions of this plan are shown in figure 4.5.

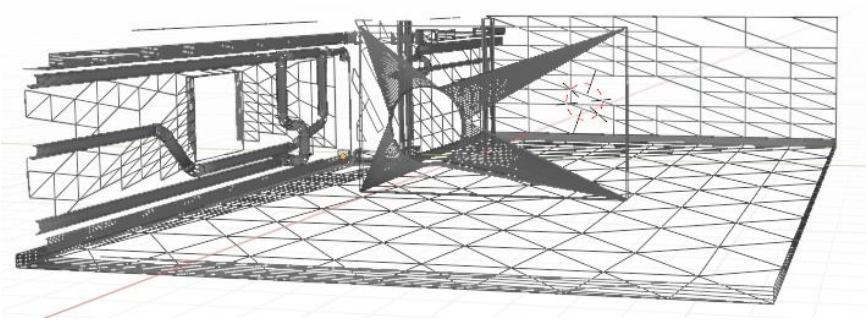


Fig. 4.4 mesh for covered vertices with 0.55 required acquired

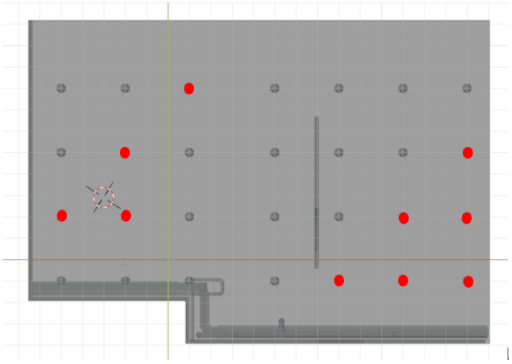


Fig. 4.5 candidate set and solution set with 0.55 required acquired

3) Required acquired = 0.5

The results, covered vertices and solution set are in figure 4.6, 4.7 and 4.8 when required acquired rate is 0.5.

```
the number of scanning points of this plan is 7
positions for scanning points in the solution set are
x:    -5     1     8    11    14    14    14

y:     2     8    -1    -1    -1     2     5

the acquired rate of this plan is 0.518256
the data overlap status of this plan is 2.109445
the computation time of the optimization algorithm is 117.921875
greedy best-first algorithm completed
```

Fig. 4.6 greedy best-first, required acquired rate=0.5

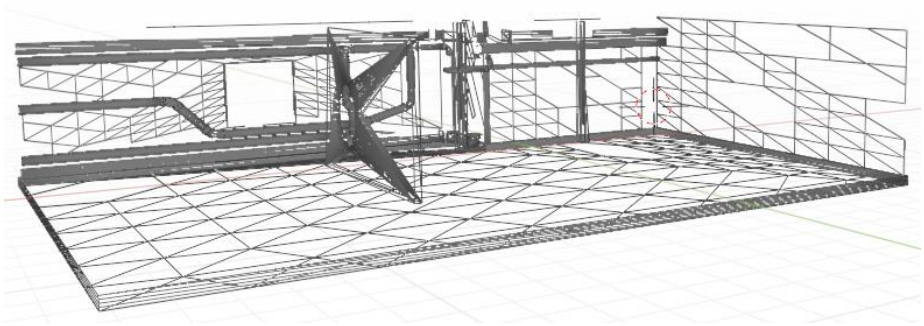


Fig. 4.7 mesh for covered vertices with 0.5 required acquired

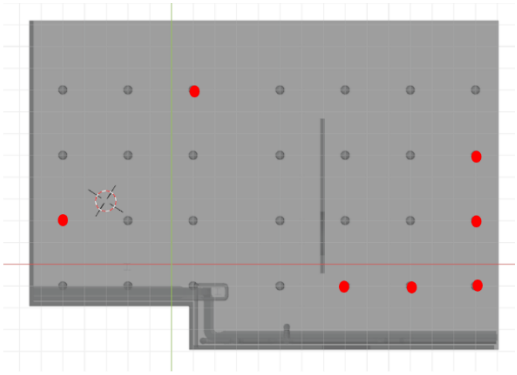


Fig. 4.8 candidate set and solution set with 0.5 required acquired

The results of the greedy best-first algorithm with 3 different required acquired rate are shown in Table 4.1.

4.1 Results and evaluations of scanning plan

Method	Optimization algorithm	Number of candidates	Required acquired(%)	Number of solutions	Acquired(%)	Overlap status	Computation time(s)
Grid-based	Greedy best-first	28	90	28	60.5949	5.0863	327.5938
Grid-based	Greedy best-first	28	55	10	55.5749	2.6775	162.5312
Grid-based	Greedy best-first	28	50	7	51.8256	2.1094	117.9219

5 Possible further improvement of the project

5.1 Generating candidate scanning positions

5.1.1 Triangulation-based method

In this task I generate the candidate scanning positions using the grid-based method. Usually another method, triangulation-based method is also widely used during the scan planning progress. For further improvement the candidate scanning points could be created by different methods to find an optimized mesh of combination of grid-based and triangulation-based with proper size, which could cover as much as information of the model with a minimized number of scanning positions.

5.1.2 Scanning route calculation

For a more complicate model there will be obstacles, also taking the scanner to different positions will be time-costing. So a route calculation and optimization method could to be implemented, in order to avoid obstacles and save shifting time.

5.2 Scanning information optimization

5.2.1 Polygons

In the project now I use vertices to as unit geometric element to present the scanning information of the model accessed by a scanning point. For further development not indices or edges, but polygons could be the unit for scanning data with respect to a scanning position, which will be closer to the real data that a scanner could get. And there all several different algorithms to check whether a polygon is visible to a certain point.

5.2.2 Occlusion Maps

The normal method to calculate the visibility for each vertice, edge or polygon and get a PVS (potential visible set) needs a large number of calculation. A possible way to optimize the algorithm is to use an Occlusion Map algorithm to get the outlines of an object and track visible points instead of storing the visibility of the whole mesh to reduce the calculation of PVS.

5.3 Improving optimization algorithms for scan planning

5.3.1 Improvements of optimization algorithms

Possible improvements could be applied for the two kinds of algorithms used in this project. For the greedy search algorithm, a backtracking process could be added on, which implements a checking process to remove unnecessary points each time a new candidate scanning point is added into the solution set for greedy best-first algorithm.

For the simulated annealing algorithm, the algorithm could remove a randomly selected candidate scanning point to find the near-optimal configuration for each position instead of removing each of the candidate scanning point and find the optimal solution.

5.3.2 Fitness function

A further improvement of the optimization algorithm is to use a fitness function to evaluate a scan plan.

$$f(S) = r \sum_{i=1}^{m} L_i$$

where S is a given scan plan, L is the length of captured building-line segments, and r is the data-overlap status. A highest possible fitness value represents a better scan plan.

6 Acknowledgement

The completion of this project is attributed to many people's support. First and foremost, I want to extend my gratitude to my supervisors, Florian and Yuandong. It's their valuable suggestions and patient guidance that help me accomplish the project. Then, I want to appreciate all the families and friends who have given me advice and encouraged me during the working period, especially Xingzhou and Lan, who have given me very useful suggestions for usage of software and visibility check, also Lay, whose spirits have inspired me a lot to work by myself. Last but not least, I would like to express my thanks to all the professors, supervisors and colleagues to attend the presentation and discuss all the topics together, which helps me to understand the project better.

7 Useful links

In the following some online sources I have relied on during my work is collected and thematically ordered. This collection is meant to provide an entry point for those who might want to improve on this project later.

7.1 Scan planning and BIM

[1] https://www.facebook.com/3DSTechnologies/photos/p.173195773365969/173195773365969/?type=1&theater.
[2] https://www.e-architect.com/wp-content/uploads/2014/12/LaserScanningtoBIM.jpg.
[3] https://de.wikipedia.org/wiki/Building_Information_Modeling.
[4] https://view.tridify.com/bim/#/conversion/M8fgGNu3TPkT0-ingY_Dj8yto2KeLjcCtGfH4BfRTws.
[5] http://www.naic.edu/~phil/hardware/laserscanner/scanstationp40p30SystemFieldMan.pdf.
[6] https://leica-geosystems.com/products/laser-scanners/scanners/leica-scanstation-p40--p30.

7.2 Visibility check and occlusion culling

[1] https://docs.unity3d.com/560/Documentation/Manual/GettingStarted.html.
[2] https://go.tridify.com/support/how-to-import-bim-models-into-unity.
[3] https://docs.unity3d.com/560/Documentation/Manual/OcclusionCulling.html.
[4] https://learn.unity.com/tutorial/working-with-occlusion-culling#5fe2b352edbc2a10f945f216.
[5] https://zhuanlan.zhihu.com/p/266592981.
[6] https://blog.csdn.net/gaojinjingg/article/details/103509420.
[7] http://ma-yidong.com/2018/02/21/some-methods-to-occlusion-culling/.
[8] https://www.cnblogs.com/sevenyuan/p/5283292.html.

7.3 Blender and bpy

[1] https://blenderartists.org/t/how-import-ifc-into-blender-without-add-on/1179144.
[2] http://ifcopenshell.org/ifcblender.
[3] https://docs.blender.org/api/current/bpy.types.Object.html?highlight=matrix_world#bpy.types.Object.matrix_world.
[4] https://docs.blender.org/api/current/bpy.types.Object.html?highlight=ray%20cast#bpy.types.Object.ray_cast.
[5] https://docs.blender.org/api/current/bpy.types.Mesh.html?highlight=polygon#bpy.types.Mesh.polygons.
[6] https://blender.stackexchange.com/questions/115285/how-to-do-a-ray-cast-from-camera-originposition-to-object-in-scene-in-such-a-w.
[7] https://www.bilibili.com/video/BV17z4y1S7nY?from=search&seid=13510257412008431211&spm_id_from=333.337.0.0.
[8] https://github.com/debaditya-unimelb/BIM-Tracker.git.

7.4 Python and Matlab syntax

[1] https://ww2.mathworks.cn/help/matlab/ref/importdata.html?searchHighlight=importdata&s_tid=srchtitle_importdata_1.
[2] https://stackoverflow.com/questions/11160939/writing-integer-values-to-a-file-using-out-write.
[3] https://cmdlinetips.com/2012/09/three-ways-to-write-text-to-a-file-in-python/.
[4] https://blog.csdn.net/qq_38422317/article/details/104499098.
[5] https://www.cnblogs.com/math98/p/10883842.html.
[6] https://jingyan.baidu.com/article/ae97a646111515bbfd461d24.html.

Reference

[1] Ernesto Frias, Lucia Diaz-Vilarino, Jesus Balado, Henrique Lorenzo. From BIM to Scan Planning and Optimization for Construction Control. Remote Sensors.2019, 11,1963.
[2] Afrooz. Aryan, Fr´ed´eric. Bosch´e, Pingbo. Tang, Planning for terrestrial laser scanning in construction: A review. Automation in Construction 125 (2021) 103551.
[3] L. Díaz-Vilariño, E. Frías, J. Balado, and H. González-Jorge, “Scan planning and route optimization for control of execution of as-designed bim,” International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, vol. 42, no. 4, pp. 217–224, 2018, doi: 10.5194/isprsarchives-XLII-4-143-2018.
[4] H. Kabir Biswas, F. Bosché, and M. Sun, “Planning for Scanning Using Building Information Models: A Novel Approach with Occlusion Handling,” 2015.
[5] Meida Chen, Eyuphan Koc, Zhuoya Shi, Lucio Soibelman. Proactive 2D model-based scan planning for existing buildings. Automation in Construction 93(2018) 165-177.