Reconstruction of Architectural Buildings from Incomplete Point Cloud Data

Master Thesis Report Stage - II

submitted in fulfilment of the requirements for the degree
Master of Technology

By

Rahul Mitra
Roll No. 123050038

Under the guidance of

Prof. Parag Chaudhuri And Prof. J. Saketha Nath

Department of Computer Science and Engineering,
Indian Institute of Technology Bombay
July, 2014

Acknowledgements

I like to express my sincere gratitude and thanks to my guides Prof. Parag Chaudhuri and Prof. J. Saketha Nath for their constant encouragement and guidance. They has been my primary source of motivation and advice during my entire study. I have greatly benefited from the complementary guidance of both my advisors on this seminar. I would like to thank Pratik Jawanpuria for his help in creating my accounts in the servers and also allowing me to access his account for installation of useful softwares.

Contents

List of Figures

Abstract

Data obtained from scanning with range scanners or obtained by Structure From Motion(SFM) from multiple images suffer incompleteness due to occlusion or lack of correspondences. This problem is significant for when large Architectural Structures scanned using *LIDAR* scanner or large scale reconstruction using SFM. So, the objective behind this project is to build a reconstruction mechanism that can predict missing structures from an incomplete point cloud of a building. Doing a surface reconstruction is time consuming and also will not work with missing data. So, in this project a modelling system is presented which perform a primitive approximation of the point cloud very FAST. The system model partial point clouds which using object recognition, which have very similar complete models in the repository.

Chapter 1

Introduction

1.1 Motivation And Objective

Data obtained from scanning with range scanners or obtained by Structure From Motion(SFM) from multiple images [2] suffer incompleteness due to occlusion or lack of correspondences. Also, sometimes the material property may have an adverse effect on the scanning process. This problem is significant for when large Architectural Structures scanned using **LIDAR** scanner or in large scale reconstruction techniques which are part of my Seminar **Large Scale 3D Reconstruction From Unstructured Images** . An example of such large reconstruction is shown in figure 1.2. So, the objective behind this project is to build a reconstruction mechanism that can predict missing structures from an incomplete point cloud of a building. In order to build a FAST reconstruction or prediction system, instead doing Surface Reconstruction (which is time consuming) which reconstruct a approximate model composed of primitive shapes. In this report we have considered Boxes as the only primitive shape. Figure 1.1 shows the schematic diagram of our project.

1.2 Overview Of the Report

In chapter 2, a part based structure recovery technique is discussed. This technique reconstructs a a mesh model from point cloud and an RGB image of an object from the same view point. It uses a repository having semantic labelled parts of models falling in the same category as the input. The technique then forms the reconstruct model by borrowing parts from different models which captures the underlying geometry of the input. The image and the scan provide complementary information. This has a similar objective to our project.

Although the above method provides visually attractive result, it requires an additional image which may not be available at all times also the method works better only for models composed only of similar parts and vary only in scale and relative position. Therefore in Chapter 3 and 4 we discuss two techniques for decomposing a point cloud into boxes(primitives). Chapter 3 provides a region growing bottom up approach that initially embeds the point cloud into a grid. Then from a staring seed(grid cell) the merges suitable cells to generate bigger boxes. The boxes found out are all axis aligned. Chapter 4 provides a top down approach which at each level splits the data when it's surface does not fits that of a cuboid. Chapter 5 provides a predictive strategy for reconstructing

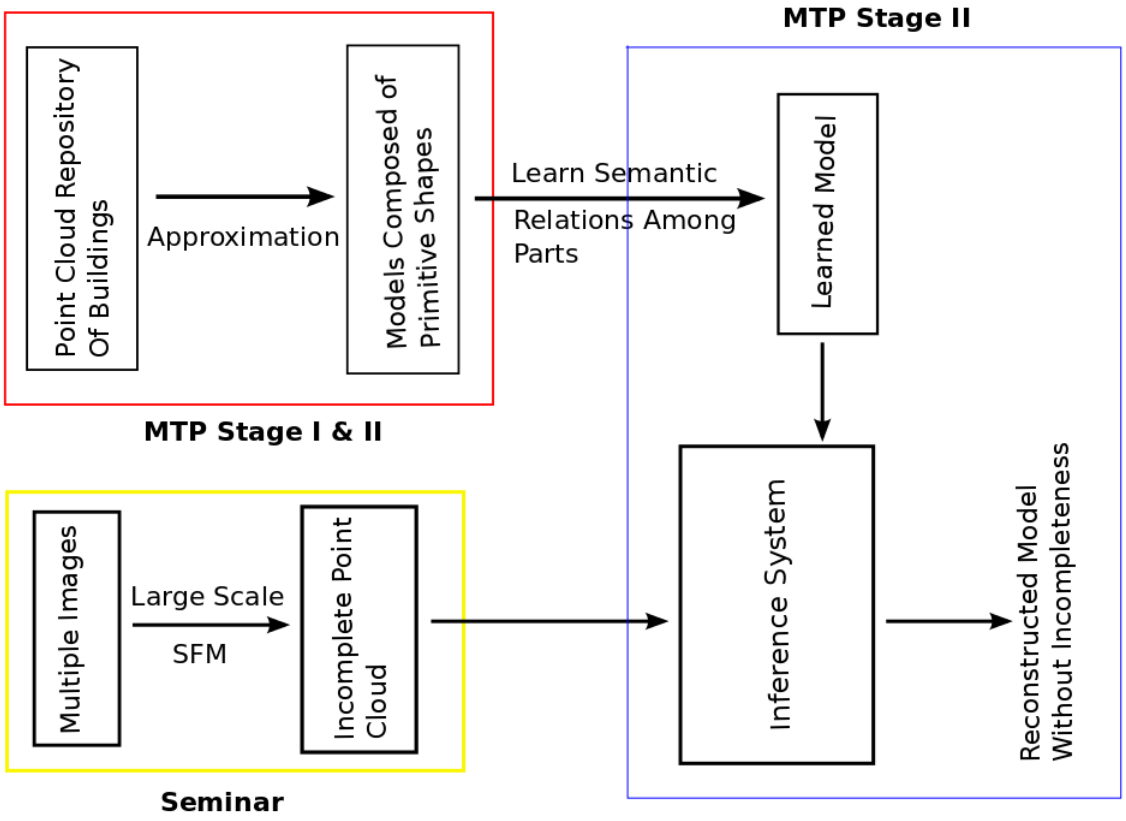


Figure 1.1: Shows the schematic diagram of our Project

incomplete point cloud along with the method described in Chapter 4.

Chapter 5 provides the Conclusion and Future Work.

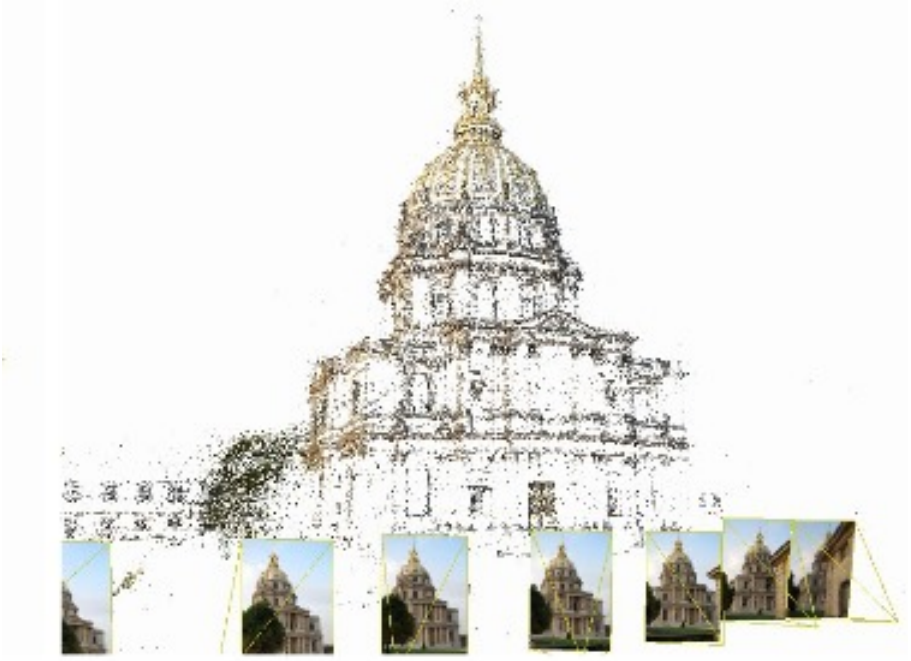


Figure 1.2: Shows a sparse and incomplete point cloud generated by SFM [1]

Chapter 2

A Part Based Structure Recovery Technique

2.1 Overview

In this chapter, a method to convert low quality point cloud data obtained from scans into a high quality 3D model with labelled semantic parts [3]. The parts are assembled in a way so that it closely resembles the underlying geometry. This approach unlike top-down approaches works bottom-up from a local to global manner which helps to keep the structure consistent with the underlying geometry. The technique mentioned assumes that most of the man made objects lie in low dimensional shape space with respect to relative sizes and positions of shape parts [4]. With this assumption, only a few top ranked candidates of certain part category are filtered out from the database and matched with acquired data. The matching considers both the partial match of the candidate part and the interaction between other parts. Then a conjoining technique is used to join the different parts into a whole.
For this technique the input consists of a single view scan of the object and an RGB image under the same view point. The reason behind taking a RGB image as input along with the point cloud is that information provided by both these sources are complementary. The point cloud may have incomplete data but provides accurate cues for the underlying geometry in 3D. On the contrary, the image have lack of depth information but have complete data of object in the current view point. The structure recovery has the following stages:

1. **Candidate Part Selection:** In this stage, for every part category a small set of candidate parts are identified without doing an exhaustive search from all possible candidate parts composition. This is done by searching for the part inside fixed window size by utilizing the fact of low shape variation. By this most parts are quickly filtered out.

2. **Structure Composition:** In this stage structures are composed by using a subset of candidate parts from each category. The composition takes into account the geometric fidelity of each part and interaction between the parts. Then a scoring function is devised to measure the individual compositions and the optimal structure is identified with the one with the highest score.

3. **Part Conjoining:** In this stage loosely placed parts from the previous stage are conjoined to form a connected structure. The proximity between the parts and and

knowledge of contact points of the different parts are used in this process.

2.2 Detail Methodology

2.2.1 Pre Processing

In this step all the models in the database are pre segmented into semantic parts and are labelled by a learning based algorithm [5]. Also reflective and rotational symmetry is detected between the parts of the individual model by [6]. This pre-processing task is done only once for each model category. In order to utilize the information from the RGB image, user interaction is required. The user can cut out the object in the image by using the grab cut algorithm [7]. Also, since the the 3D scan and the image are taken from the same view point, by projecting parts of the object in the image we can extract its corresponding 3D scan pieces from the rest of the scanned data.

2.2.2 Candidate Part Selection

The selection of the candidate parts from the database is based on the degree to which the part matches with both parts of the scan and the image. The parts from each model in the repository are matched efficiently by a **two** stage process.
First a rough matching of each individual repository model with the input data is done. To do this, first individual models aligned with scan data. The scan data and the repository model are aligned by first aligning them with their upright direction. The upright direction of the the repository models are already pre-defined and fixed. The upright the scan data can be determined by detecting the dominant supporting plane [8]. After this, each model is scaled and translated so that it can just fit within the Bounding Box of the scan. Next, the orientation of both the entities around the upright axis are made same. This is done by rotating the scan cloud by some sampled rotation and checking for the min squared error. Figure 2.1 shows models of chairs aligned with scan of a chair. Next individual parts of the aligned model are matched with the scan data. The RGB image is also used for matching with the projection of the repository model and a final score is computed for each model. An example for the chair scan is shown in figure 2.2. Here the image information is used to match the parts when scan data is missing as shown in figure 2.3.

2.2.3 Structure Composition

In this step, the underlying structure is composed by taking a subset of parts from each category(empty or 1) by following some constraints:

- **Geometric fidelity**: The parts of the composition should match the acquired data well i.e good geometric fidelity score.
- **Proximity**: The parts should not be far away from each other.
- **Overlap**: The parts should not overlap with each other too much.

2.2.4 Part Conjoining

After completion of the previous step the parts selected are loosely placed together. This step conjoins all the parts to form a well-connected model. First pairs of parts that are

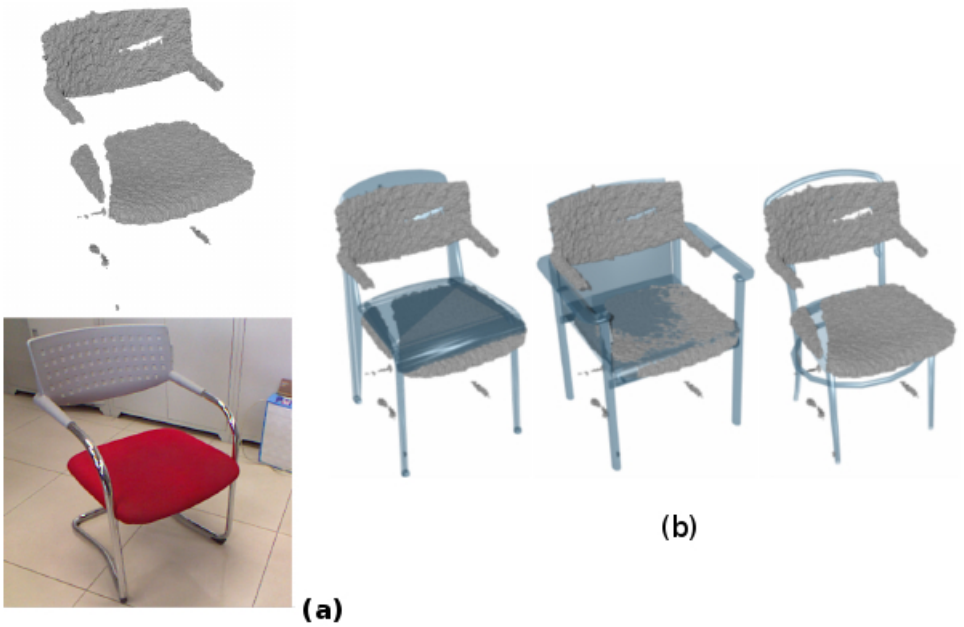


Figure 2.1: (a) shows the scan and the RGB image of chair whose scan is to recovered. (b) Shows orientation of the models from the repository after alignment with the point cloud [3].

likely to contact each other are found out. For this proximity between the two parts is used as a measure to select a pair. However only using using proximity does not yield a good result. So, the repository is used to infer whether the two parts should be contact or not. Figure 2.4 shows the contact points of different parts in the repository.

After pairing of the contact points, the sizes and positions of the parts are adjusted so that pair of contact points meet each other as much as possible.

2.2.5 Results

The database had 70 models of each category mainly chair, tables, bicycles and aeroplanes. with labelled parts. The value of $\mathbf{K}$ i.e number of top ranked parts from each category is fixed to 5. The spatial overlap threshold parameter $\mathbf{t_s}$ is decreased when excessive tiny parts are to be avoided. The geometric fidelity threshold $\mathbf{t_f}$ is decreased when the scan data is very poor. The distance threshold $\mathbf{t_d}$ is increased to use more primary primary contact points for conjoining. Some of the results are shown in figure 2.5 and 2.6.

This approach is view independent and generated roughly the same structure for different view scan of the same object. Limitations of this approach:

- Requires an extra RGB Image to extract information not present in the scan data.
- Works efficiently for objects which lie in low dimensional shape space.
- This method fails for objects fails for objects which features that cannot be characterized by spatial layout e.g. façades.
- In-correctness in creating the edge map where 3d scan data is not available can cause match with undesired parts.

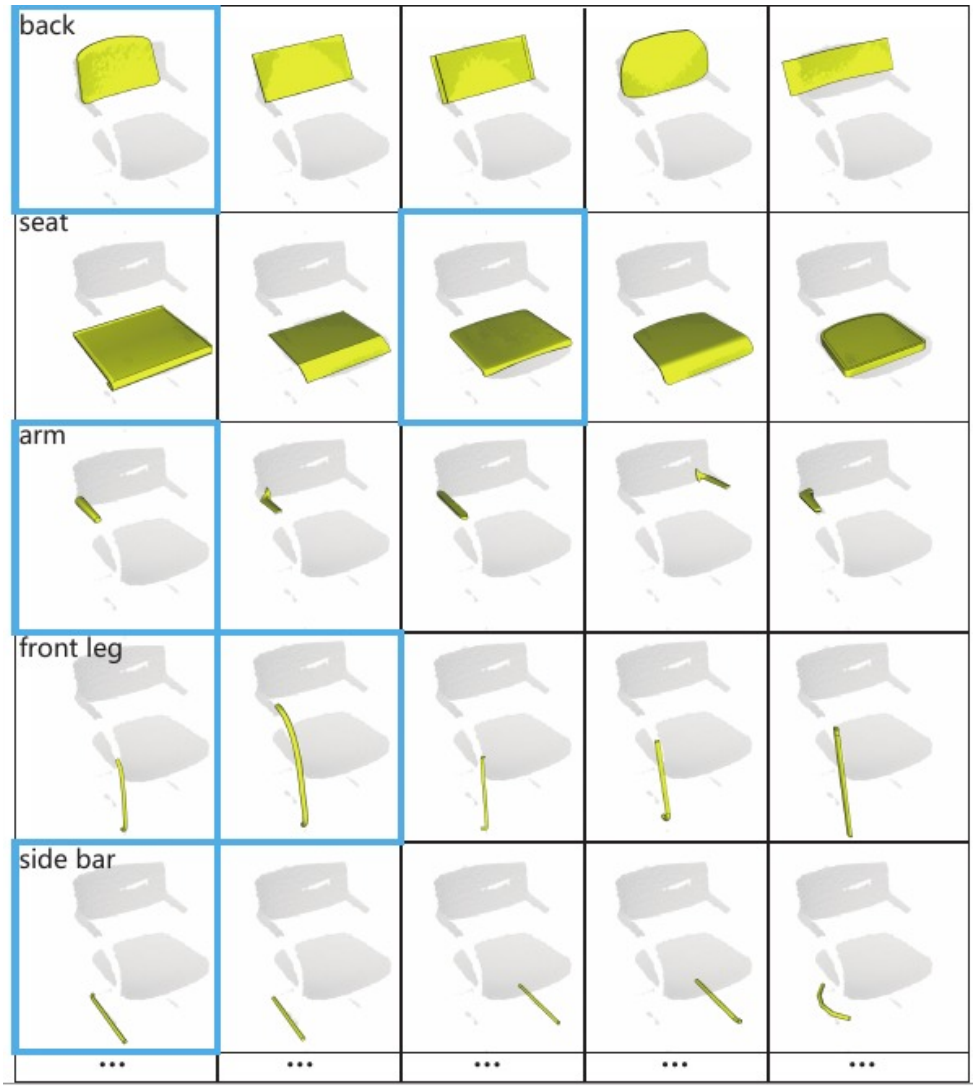


Figure 2.2: Shows the top K(here 5) candidates for each category, the blue box denotes the part that are finally selected after all the steps are processed [3].

- For low resolution input data parts very close to each other cannot be discriminated accurately as shown in figure 2.7.

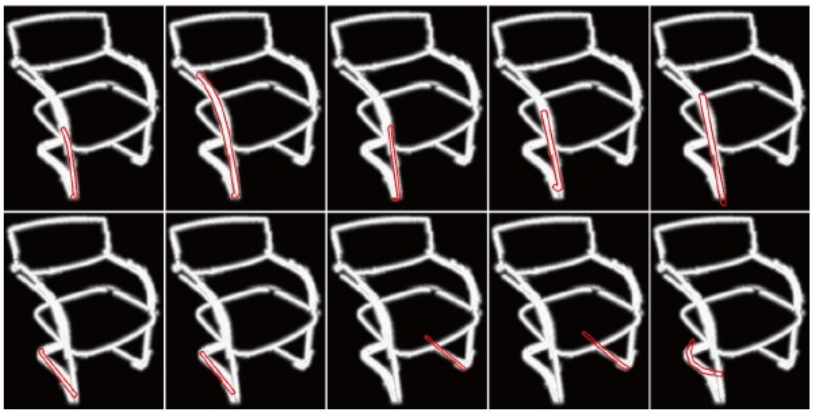


Figure 2.3: Shows 2D Distance field image is used to match parts from the repository for the leg of chair whose data is missing in the scan [3].

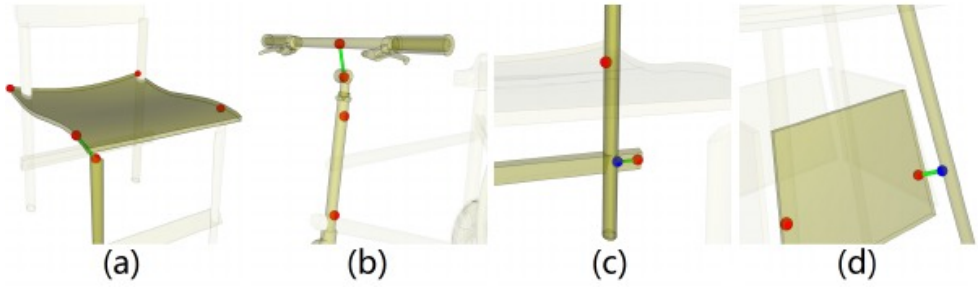


Figure 2.4: The red points shown above are the primary contact points and the blue points are the secondary contact points [3].

Figure 2.5: The above figure shows the final reconstructed model from scan data by parts borrowed from different repository models. Here the H shape base type example did not exist in the repository but recreated from other parts [3].

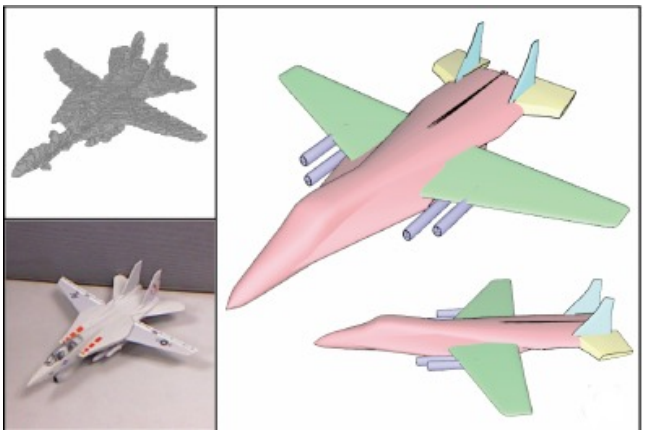


Figure 2.6: The above figure shows the final reconstruction from scan of plane model. Different colored parts are taken from different model. From the figure we can see symmetry is used for generation of symmetrical parts [6].

Figure 2.7: The above figure shows the failure of the part matching algorithm to discriminate between 3 slim pillars that are close to each other[3].

Chapter 3

A Region Growing Approach To Box Decomposition of Point Cloud Technique Using Axes Parallel Boxes

3.1 Overview

In this chapter we discuss a bottom up region growing based technique which decomposes a point cloud into axis aligned boxes. This method is inspired from [14] which uses a bottom up merging technique to build **LEGO** models of objects.
In this method, the initial step is to embed the 3D scan data into a voxel grid. The voxel grid is always axis aligned and the dimensions of the grid are found out by calculating upper and lower bound of the data. The cells in this grid instead of being a cube is a cuboid whose dimensions are a pre-determined fraction of the grid dimensions. After this, cells of the grid that does not intersect with the scan data are removed. Then cells are merged with neighbouring cells which are **compatible** only **X** direction. The idea of being compatible is that after merging the merged cell should be also in form of a cuboid. Next, the resulting merged cells re merged in the **Y** direction and then in the **Z** direction. At any point of time in the above process a box composed of many cells is represented by the co-ordinates of the cell has the lowest **Z**, **Y** and **X** value as shown in figure 3.1. Since it is Box, we can always find a cell having the lowest values in all the 3 directions. To implement the concept of region growing, all the valid cells are stored in an ordered-map ,sorted by their **Z**, **Y** and **X** value in order. So, in the merging phase, since the cells are kept sorted, adjacent cells if present are merged. In this way a region continues to grow in a direction in a plane until there are no more cells to merge. This is done for all planes sequentially in order of increasing perpendicular distance from the staring plane as shown in figure 3.2. The merging is done sequentially for all the three directions. After merging in all the three directions, it results in the minimum number of axis parallel boxes required to represent the voxel grid.

3.2 Detailed Algorithm

Each cell in the voxel grid contains, a counter **dataCount** and boolean variable **isNoise**. The outline of the voxelization process is given as follows,

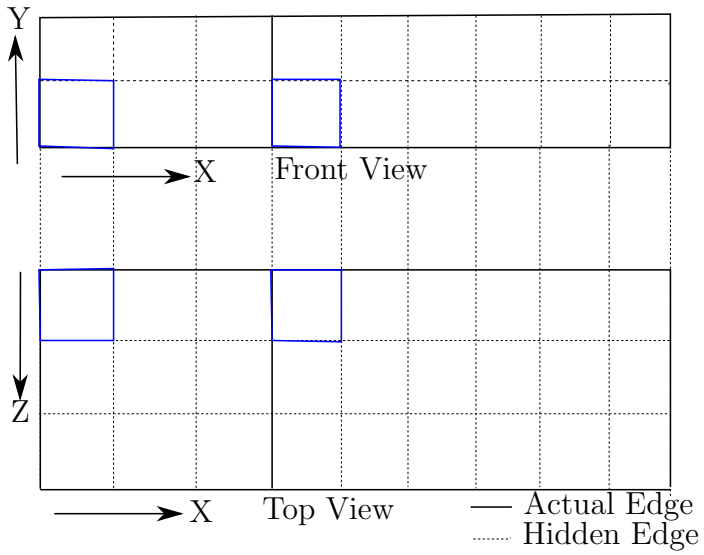


Figure 3.1: The above figure shows the Front and Top View of two compatible boxes in the **X** direction. The boxes are compatible as they have both **Y** and **Z** values of the representative cell equal. Also the dimension in **Y** and **Z** are equal. The blue boxes shows the representative cell of each of boxes. After merging the resultant box will have the representative cell as the left blue box as it has a lesser **X** value of the two boxes.

Algorithm 3.1 Voxelization of scan data

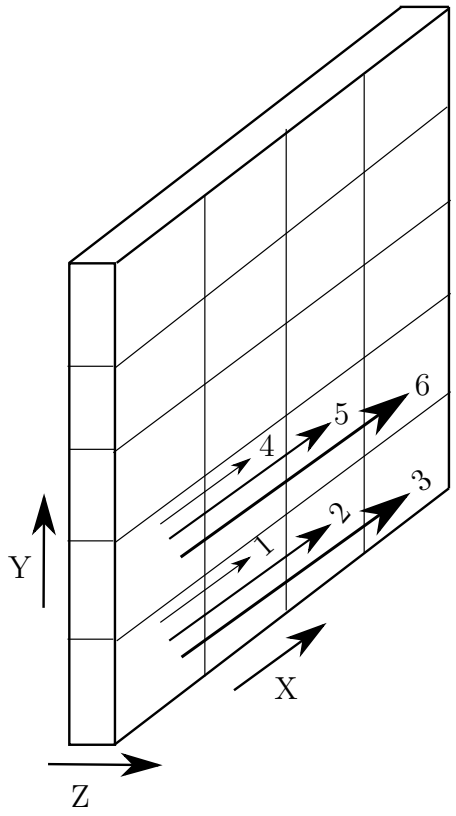
Input: Point Cloud Data **D**
Output: Grid **G**
Step 1: A axis parallel grid **G** is formed by computing lower and upper bound of data.
Step 2: The dimensions of a cell in the grid **G** is calculated by taking a pre-determined fraction of the grid size. In the experiments the value is taken as 0.2% of the size.
Step 3: For each data point in cloud, its nearest voxel co-ordinate is determined and its **dataCounter** in increased.
Step 4: Voxels having data points less than a threshold, its **isNoise** value is set to false;
Step 5: Voxels having **isNoise** set to true are stored in a ordered-map(**M**), ordered by **Z**, **Y** and **X**.

Once the grid and the ordered-map(**M**) has been created, the merging process is started. Merging algorithm for a fixed direction(**X** is example) is given below,

Algorithm 3.2 Merging cells in one direction

Input: Grid **G**, ordered-map **M**
Output: Grid **G**, ordered-map**M**
Step 1: Take **C** to be the first element of **M**.
Step 2: If any of the adjacent Box(**B**) of **C** in the **X** direction has the same textbfY and **Z** co-ordinate the two Boxes are merged in Step 3 and 4
Step 3: Find out which out of the two Box comes ahead in sorted order. Merge the Box which comes after in the order with the one which comes before.
Step 4: Remove the one which comes after from **M**.
Step 5: If a merge occurred in Step 2, then goto to Step 2.
Step 6: If atleast once a merge occurred in Step 2 then goto Step 1.
Step 7: Return.

The algorithm 3.2 is called 3 times sequentially with merging occurring in the **X**, **Y** and **Z** direction as shown in figure 3.2. Due to poor quality of surface, there may be gaps in between voxels even for a continuous surface. Such gaps may lead to a break in region growing process and leads to unnecessary increase in the number of boxes. A tweak in algorithm can reduce the number of boxes in such a case by not only considering the adjacent box but considering any box within a fixed distance.



Plane with Z = i

1, 2,...,6,.. is the sequence in which merging is going to take place while merging in X direction.

Figure 3.2: The above figure shows a XY plane with Z = i. The merging algorithm first along the X direction for a given Z and then in the next iteration Z is increased the same procedure is followed. Similar situation happens when merging along Y and Z axis.

3.3 Results

This section shows the results on running the algorithm on architectural structures downloaded from **Google Warehouse**. The models available in **Google Warehouse** rare not point clouds but mesh models and are available in **collada** format. Point cloud of such models are created by sampling the meshes using **Poisson Disk Sampling**. **Point Cloud Library** is used to load and process the ply file. Figure 3.3, shows the result of running this algorithm on different models. Here the initial cell size is taken as **0.05** of

Table 3.1: Shows information about the Box Decomposition of different models in figure 3.3

Model	Points	Initial Cell No.	Box No. After Merging	Total Time Taken
Fig. 3.3 (a),(f)	1268039	1879	42	6.25 secs
Fig. 3.3 (b),(g)	469003	674	34	2.16 secs
Fig. 3.3 (c),(h)	1030023	1645	48	4.86 secs
Fig. 3.3 (d),(i)	977029	310	7	4.48 secs
Fig. 3.3 (e),(j)	2223314	2325	96	10.58 secs

the bounding Box. Threshold for noise while voxelization is kept at 4 points.

This algorithm works very fast in practise, to generate the voxel grid one iteration of through the point cloud data is required i.e. O(N) where N is the number of points. Since the geometric information about each voxel is ordered map, it takes $O(\log V)$ time to access the information of a voxel where V is the number voxels present in the map currently. So, the merging phase takes O(VlogV) to execute.

The shortcoming of this approach is that it performance degrades with poor quality data. Also for surfaces that are not parallel to the axis, the approximation is poor as shown in figure 3.3 for (c) and (h). Also curved surfaces are approximated poorly by boxes. Also to connect cells that are separated by gaps due lack of data the condition of compatibility between the boxes is relaxed. But this also sometimes leads to connecting different parts as shown in 3.3 (d) and (i). Table 3.1 provides some information regarding the results obtained in figure 3.3.

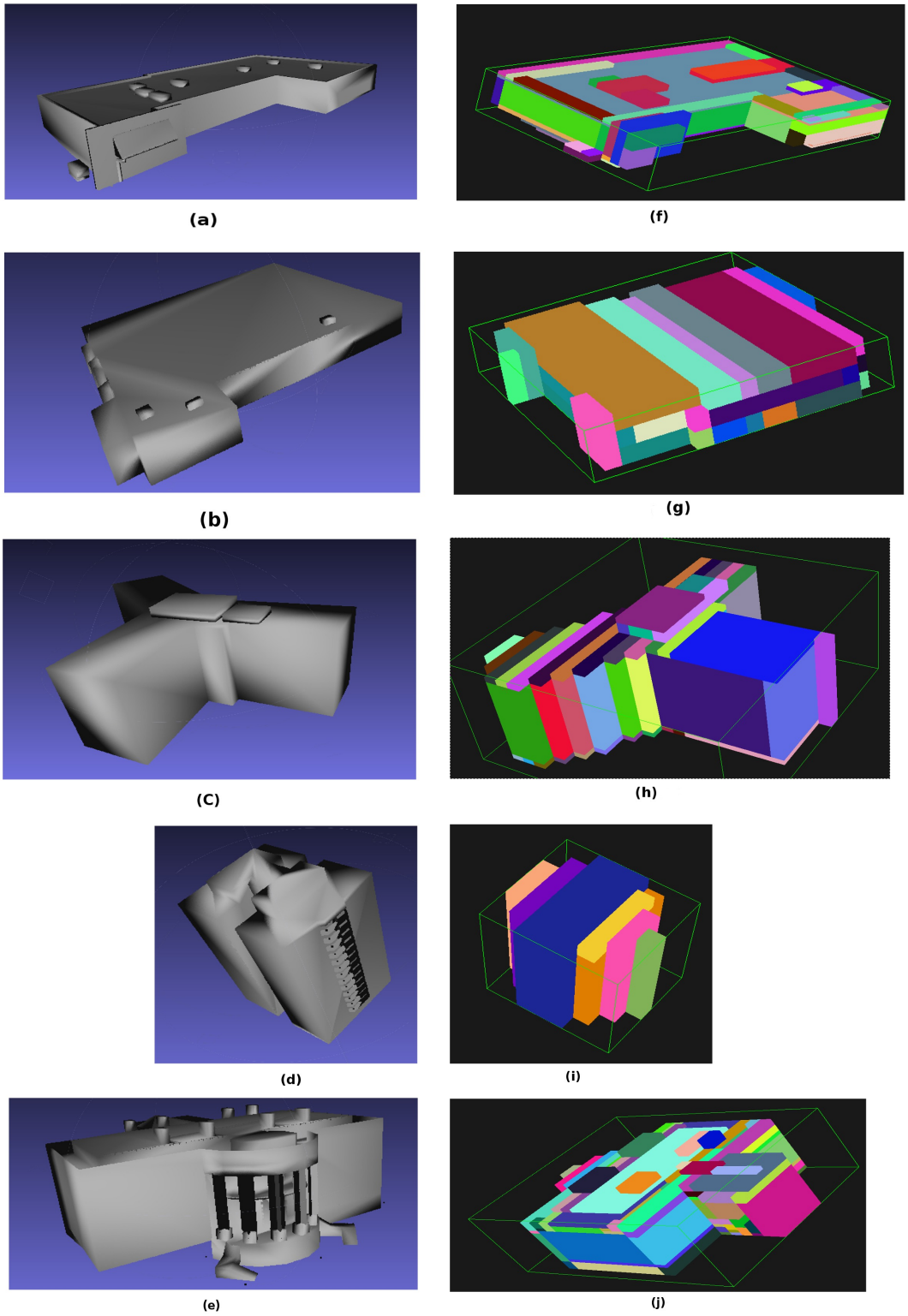


Figure 3.3: In the above figure (a)-(e) shows the point cloud data of the different buildings downloaded from Google Warehouse. Figure (f)-(j) shows the axis parallel box decomposition of the corresponding point clouds

Chapter 4

A Top Down Approach For Approximating of Point Cloud Data Into Boxes

4.1 Overview

This chapter shows a Top Down technique for decomposing a point cloud into minimum number of axis parallel boxes. The idea behind this approach is inspired by [11]. Unlike the first approach mentioned in the previous chapter, representing a model by boxes can be done automatically and user interaction and repository of labelled semantic parts of a model is not required. Also, unlike region growing approaches. Also this technique suffers less when data is noisy or when the scan quality is poor as experienced by bottom up region growing methods [12].

4.2 Detailed Algorithm

Here first an initial bounding box is calculated. Then the data inside the box is checked for its RMSE from the surface the box. If the RMSE is more than a pre-determined threshold the box is splited further. The splitting is done along one of the co-ordinate axis and determined maximum reduction in volume caused by the split. The reduction if at all is there due to the fact that the volume covered the bounding boxes of two parts of the splitted data is less equal to that of the parent bounding box. As with the previous merging approach the point cloud is first embedded into a voxel grid.

The outline of the algorithm is as follows,

Algorithm 4.1 Box Decomposition Outline

Input: Point Cloud Data $\mathbf{D}$
Output: Set of Boxes each bounding part of the Point Cloud Data
Step 1: Find the bounding box $\mathbf{B}$ for the current data $\mathbf{D}$ by min-max of the data.
Step 2: Check whether the data inside the box has Root Mean Square distance from the surface less than a threshold. If yes goto step 3 else goto step 7.
Step 3: Find the **BEST** split parallel to any of the faces of the box and divide the data into two parts say $\mathbf{D_1}$ and $\mathbf{D_2}$ as shown in figure 4.1.
Step 4: Create Bounding Boxes $\mathbf{C_1}$ for $\mathbf{D_1}$ and $\mathbf{C_2}$ for $\mathbf{D_2}$.
Step 5: Find the corresponding reduction in volume of the split in step 3 by, RED $= \frac{\mathbf{V(C_1)+V(C_2)}}{\mathbf{V(P)}}$, where $\mathbf{P}$ is the parent box.
Step 6: If the RED computed in Step 5 is more than a usually 0.98 then repeat recursively Steps 2-5 for $\mathbf{D_1}$, $\mathbf{C_1}$ and $\mathbf{D_2}$, $\mathbf{C_2}$. **Step 7**: Store the data and bounding box of the parent.

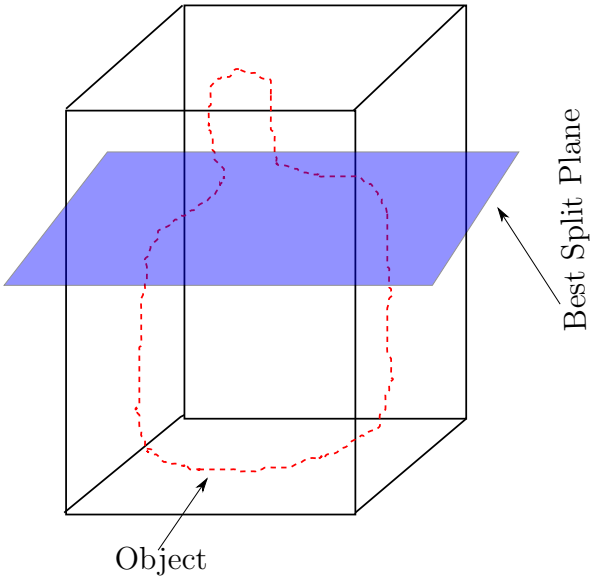


Figure 4.1: In above figure, the blue plane is the best splitting plane parallel to the top and bottom face. Points lying on top of the plane are $\mathbf{D_1}$ and bottom are $\mathbf{D_2}$. This figure is in 2D but the splitting is done in 3D.

4.2.1 Computing The Best Split

The split of a Bounding Box is done parallel to any of the face. To find the best best split for a particular cutting plane (say XY), a loop is run along the direction perpendicular to it(here it is Z direction). For every index of the grid in the Z direction reduction in volume is computed. The index where minimum reduction occurs is returned as the best split for that plane. This is done for the other two planes also. Then the minimum of the three splits is used to split the data.

4.2.2 Computing RMSE of the data

To compute the RMSE, for each data voxel (see voxilization in previous chapter) find the record the nearest distance from the surface of its bounding box. To ensure that a noisy

surface does not effect the RMSE a the nearest distance is subtracted by 1 unit while calculating.

4.2.3 Heuristic When Volume Reduction Fails

There are cases when volume reduction fails although the data has a high RMSE as shown in figure 4.2. However these cases does not occur very frequently.

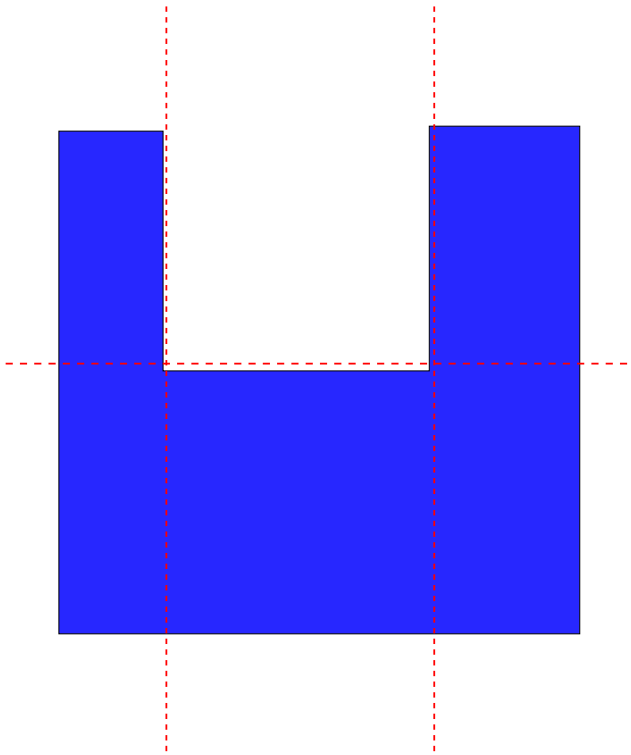


Figure 4.2: The above figure shows is a 2D projection of 3D model. That is, the projection on all the four sides should be similar to the above figure. Here there is no reduction in volume cutting through any plane parallel to the axis. However the red dotted lines show the possible cuts.

Here to find the best split for a cutting plane, we use the following algorithm based on rate of change of RMSE:

Algorithm 4.2 Heuristic To Find The Best Split For A Cutting Plane

Input: Voxel Grid and Parent Bounding Box **D**
Output: Index of the Best Split in the Voxel grid
Step 1: Loop in the direction perpendicular to the plane. For each index of the grid in that direction calculate and store RMSE of both the lower and upper block considering the parent is splitted at that index in an array **S** separately. Also store the min RMSE found for both the upper and lower blocks. To avoid edge cases we loop not from begin to end index but within a margin of 4 units from begin and end
Step 2: Loop through **S** and calculate the difference in RMSE of $S[i]$ and $S[i+1]$ and store in $\mathbf{S_{1u/l}}[i]$ for both lower and upper blocks separately.
Step 3: Calculate the index having the maximum jump in RMSE among both $\mathbf{S_{1u/l}}[i]$ but have $S[i-1]$ within a threshold of the min RMSE found in Step 1. **Step 4**: The index i calculated in the previous step is considered the best split in that direction.

4.3 Results On Complete Synthetic Data

The parameters of our the algorithm are set in the following way,

- The size of the voxels are kept at 0.1 fraction of smallest dimension of the initial Bounding Box.
- The threshold for the reduction in volume is kept at 0.98-0.99 of the initial volume.
- The RMSE threshold is kept at 0.2 units.

The number of boxes used in the decomposition depends on the threshold value of the reduction and RMSE.

4.3.1 Creation Of Synthetic Data

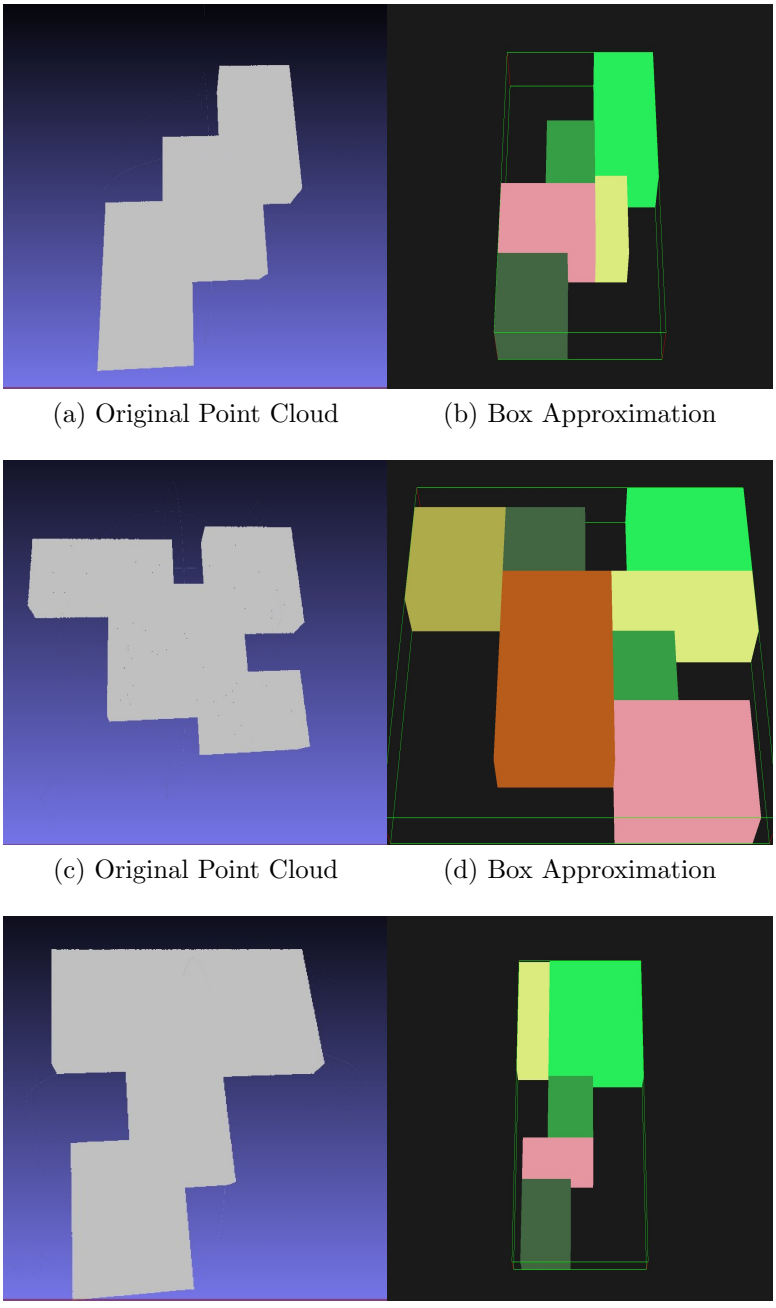
Synthetic Data is created in 3DS MAX software using the script [16]. The script is used to generate a random perimeter with perpendicular corners. Some buildings with non perpendicular walls are created manually using Blender. After creating the mesh model **Poisson Disk Sampling** is used to create the point clouds.

The figures 4.3 and 4.4 below shows decompositions obtained.

4.4 Results On Complete Models Gathered from Google Warehouse

The figure 4.5 below shows decompositions obtained,

An advantage of this approach is that after the decomposition of two separate branches can be done independently. Also more primitives can be easily incorporated in this approach.

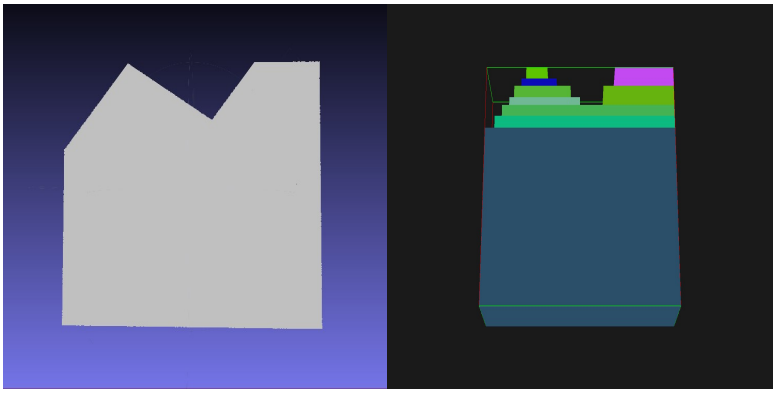


(a) Original Point Cloud (b) Box Approximation

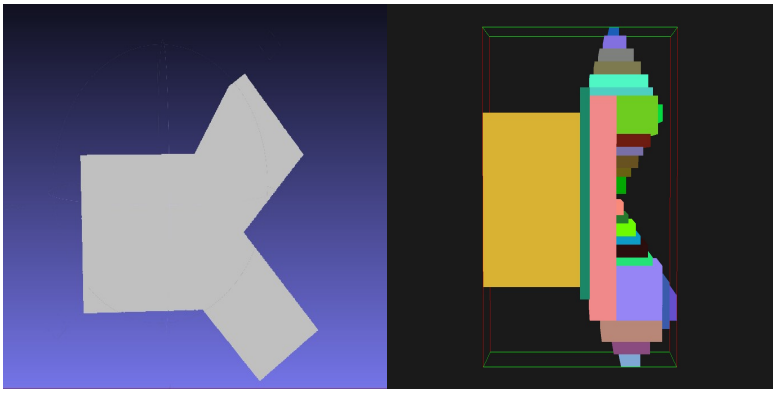
(c) Original Point Cloud (d) Box Approximation

(e) Original Point Cloud (f) Box Approximation

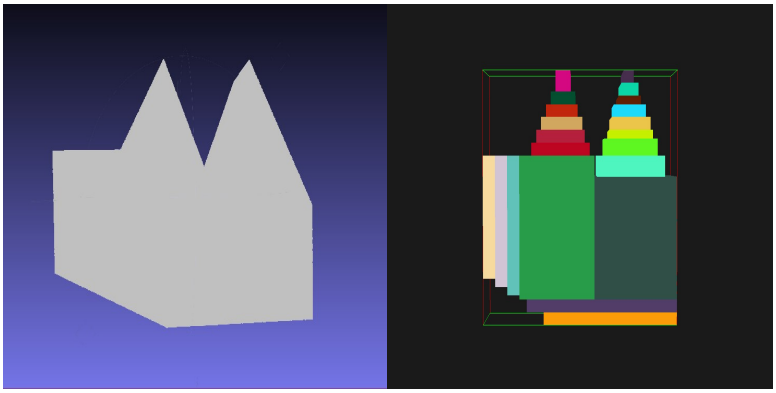
Figure 4.3: The left column shows the point cloud data and the right column shows the Box Decomposition. Here the data contains only perpendicular walls.



(a) Original Point Cloud (b) Box Approximation

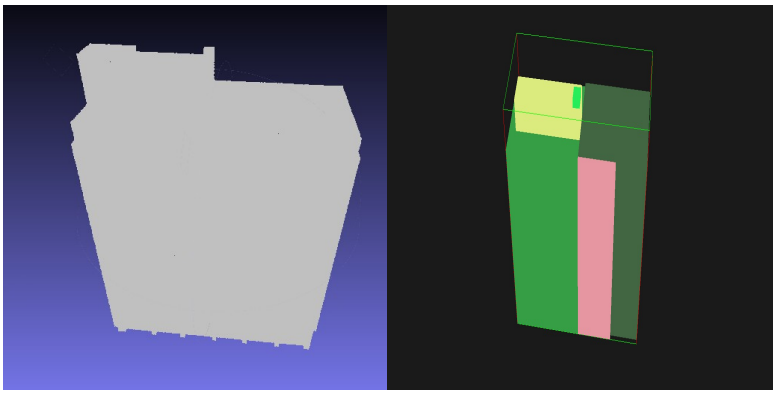


(c) Original Point Cloud (d) Box Approximation

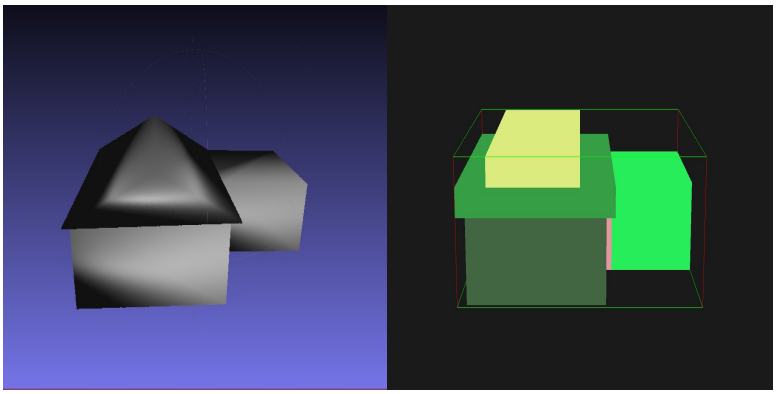


(e) Original Point Cloud (f) Box Approximation

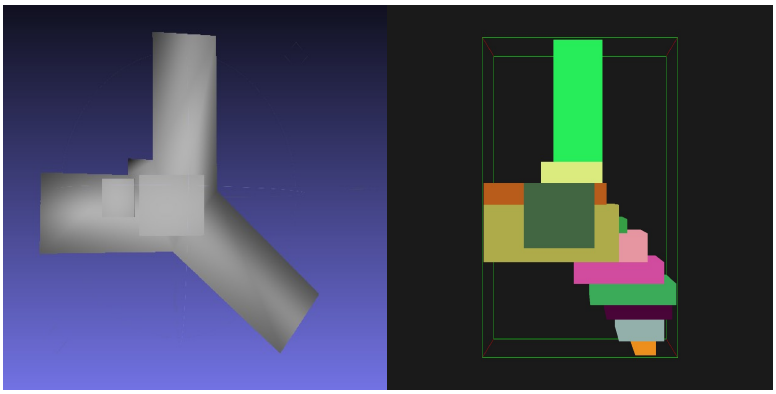
Figure 4.4: The left column shows the point cloud data and the right column shows the Box Decomposition. Here the data contains Oblique walls.



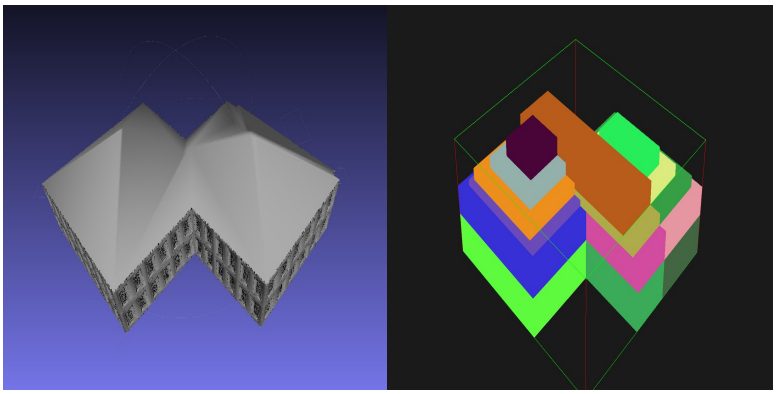
(a) Original Point Cloud (b) Box Approximation



(c) Original Point Cloud (d) Box Approximation



(e) Original Point Cloud (f) Box Approximation



(g) Original Point Cloud (h) Box Approximation

Figure 4.5: The left column shows the point cloud data and the right column shows the Box Decomposition. Here the data is obtained from Google Ware House.

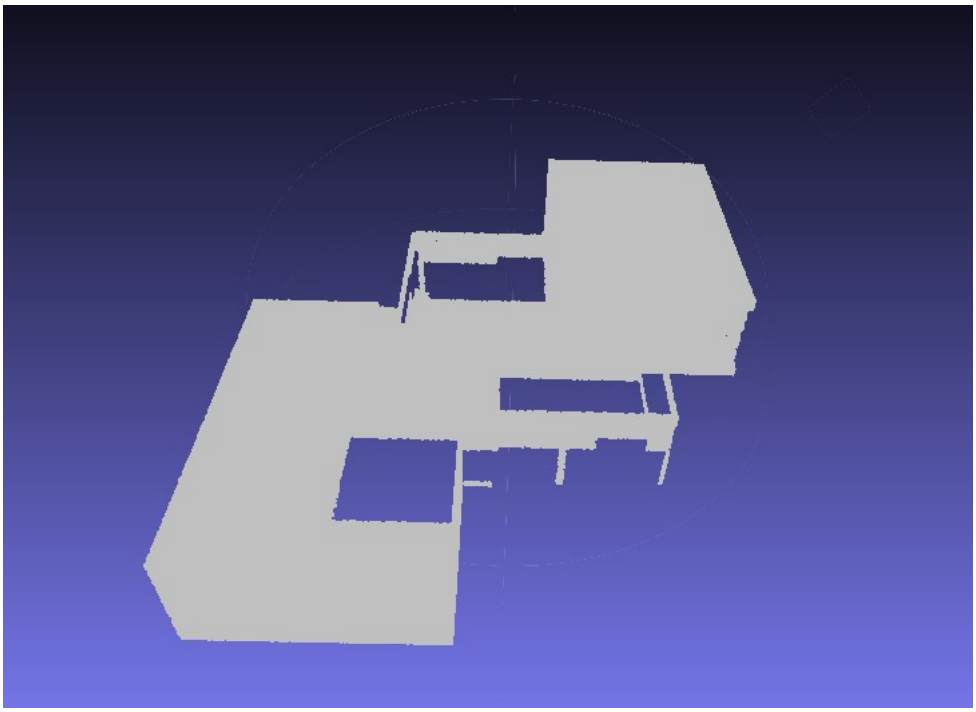


Figure 4.6: The above figure shows an partial cloud generated from Point Cloud shown in figure 4.3 (a).

4.5 Creation Of Synthetic Partial Data

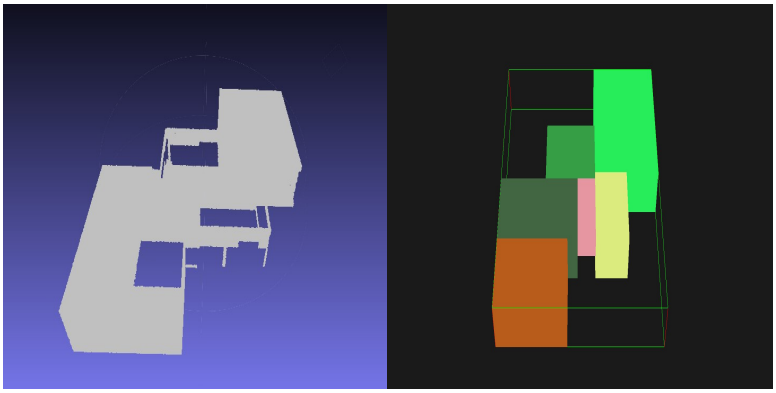
Partial Data is created by randomly removing a rectangular patch on any of the six sides of the bounding box. The co-ordinates of the rectangle are selected at random but it is ensured that none of the dimensions of the rectangle exceed more than 50% of the any of dimension of the initial Bounding Box. To do so, first one of the six side is chosen at random. Then a point(P) on its surface is chosen at random. The next three points are chosen depending on whether co-ordinates of P are more than halfway of each dimension or not not. An example of such a creation is shown in figure 4.6.

4.6 Results On Partial Data

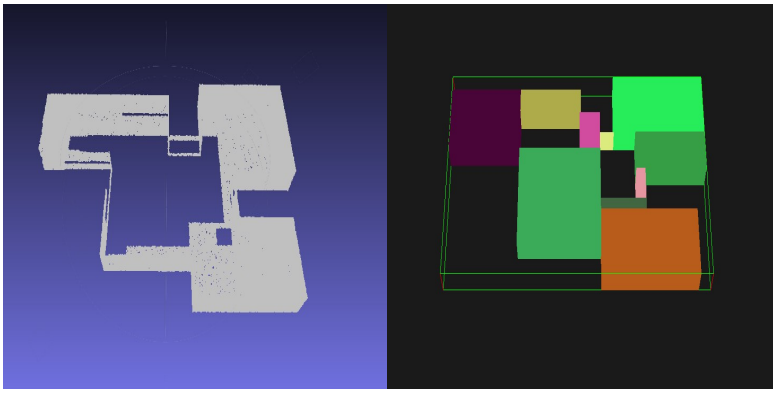
The following figure 4.7 shows decomposition obtained,

4.7 Results On Data Obtained After Applying SFM On Photographs

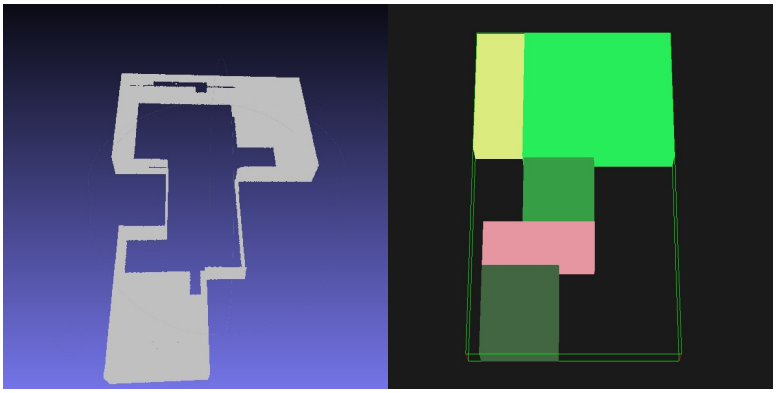
The following figure 4.8 shows the result from point cloud obtained after SFM from pictures.



(a) Original Point Cloud (b) Box Approximation

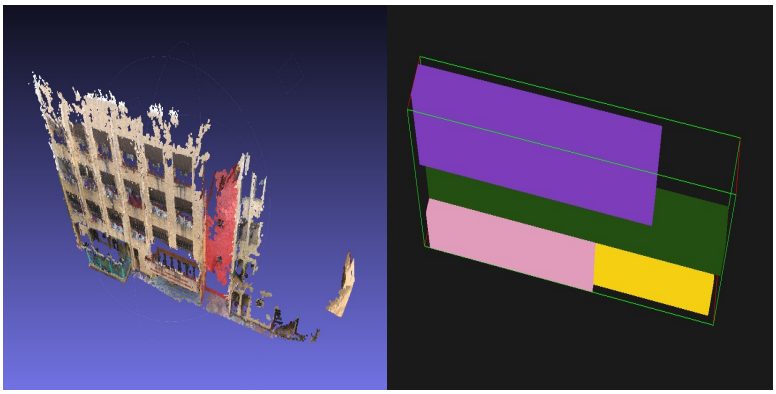


(c) Original Point Cloud (d) Box Approximation

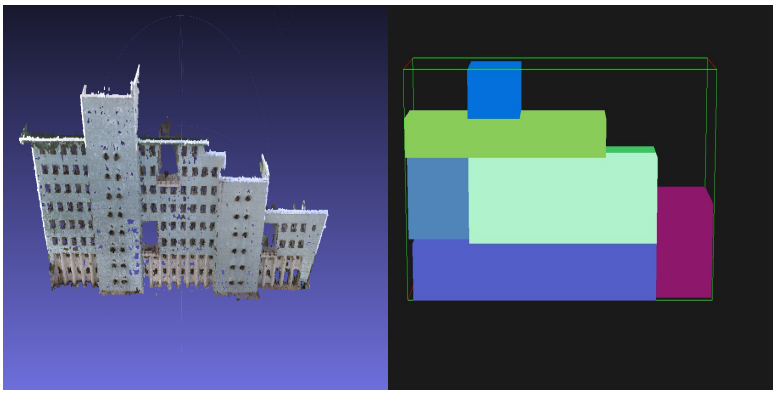


(e) Original Point Cloud (f) Box Approximation

Figure 4.7: The left column shows the point cloud data and the right column shows the Box Decomposition. Here the models have missing data.



(a) IIT Bombay SOM (b) Box Approximation



(c) IIT B Hostel 14 Wing (d) Box Approximation

(e) Cornell Art Quad Watch Tower (f) Box Approximation

Figure 4.8: The left column shows the point cloud data and the right column shows the Box Decomposition. Here the models have missing data.

Chapter 5

Using Learning Techniques For Decomposition Of Partial Data

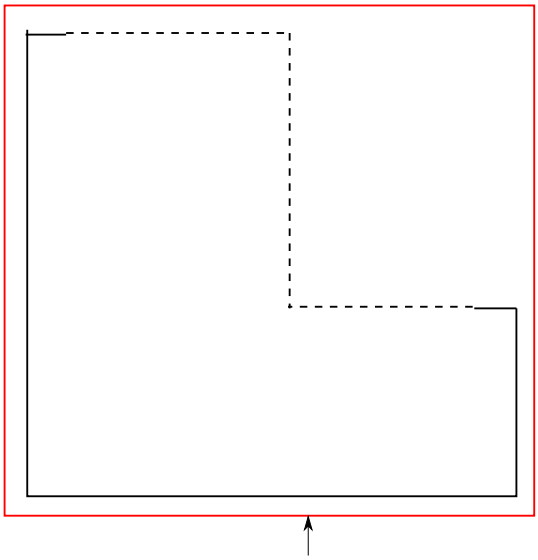
The errors in decomposition that occurs in partial data are of mainly Two types:

- Under Split: This happens when the RMSE of the data inside the box is close to ZERO due missing data of an entire side or corner as shown in figure 5.1 (a).
- Incorrect Decomposition: Due to missing data the BEST split calculated is worse than the one calculated for complete data of the same model. Also, Bounding Boxes of the two children can be incorrectly calculated due to missing data as shown in figure 5.1 (b).

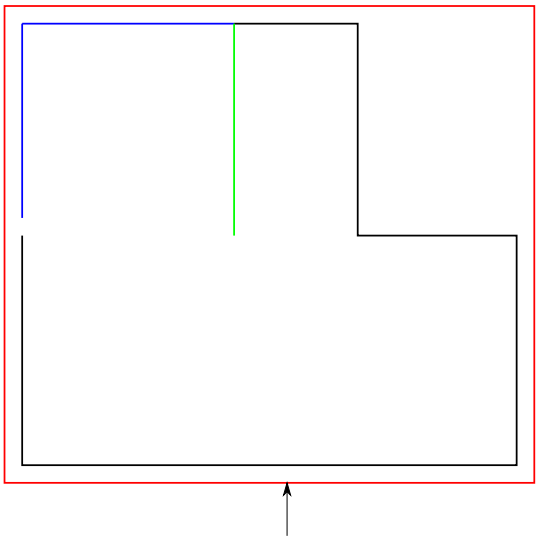
5.1 Outline For Creating Training Data

The training data is created by essentially storing the Decomposition Tree for our Training Data set consisting of 15 synthetically created models. Here for each node in the tree the following information is stored,

- Which directions are the longest(l1), second longest(l2) and third longest(l3) dimensions.
- Aspect ratios i.e. (l3) to (l1), (l2) to (l1) and (l3) to (l1).
- For each of the two child box, the offset of representing co-ordinate(mentioned in chapter 2) from the parent's representing is calculated. The ratio of the offset to its corresponding dimension is stored.
- Similarly, the 3 dimensions of the child boxes are stored as ratios of its parents corresponding dimensions.
- The name of model is stored.
- The number of the node in the tree is stored. i.e. **2i** or **2i+1** when parent has a number **i**.
- Information about whether the node is stored or not is also stored.
- The Point Cloud is stored.
- The descriptors extracted from the data is stored.



(a) Missing Data Ex. 1



(b) Missing Data Ex. 2

Figure 5.1: The Bounding Box in the above figures is drawn with one unit gap for better clarity.In (a) RMSE is close to 0 due to missing data hence no decomposition is done. In (b) Bounding Box of the top should be as shown in blue but wrongly calculated shown in green. The black solid lines shows data.

5.2 Descriptors Used

For now only local descriptors are used. Here Signature of Histograms of Orientations(SHOT) [17] feature is used. This is an 3D equivalent of SIFT [18]. Before encoding each keypoint (uniform down sampled data points in this) to SHOT descriptors, the Local Reference Frame for each keypoint point is found out as mentioned in [19] to make the descriptor invariant to rotations and translations. For encoding the descriptor the radius within which support points are to be looked is taken as 5 times the cloud resolution.

5.2.1 Matching Descriptors and Correspondence Grouping

In order to determine whether training database model is similar to the scene using SHOT descriptors, first descriptor for the each each data point in the scene are matched to the nearest neighbour of considering there distance between the histograms is below a certain threshold(0.25). After the pairs of matches are found out (total number of pairs of matches is a subset of the total number of key points) a Hough voting scheme [20] is used to a find clusters of the matching pairs that satisfies a rigid transformation from scene points to repository model points. For similar objects number of clusters is very less(less than equal to 5).

5.3 Algorithm For Testing Data

The outline of the testing algorithm is as follows:

Algorithm 5.1 Testing Algorithm Outline

Input: Scene Point Cloud Data **D**
Output: Set of Boxes each bounding part of the Point Cloud Data
Step 1: Find the aspect ratios of the current block (p).
Step 2: Find the nearest neighbour within a threshold radius(0.05 units) from the training database.
Step 3: For each of the blocks retrieved from the database match its SHOT features with that of the scene and perform Correspondence Grouping to enforce geometric constraint. The training having the minimum number of clusters and the number is within 5 is taken as matched model.
Step 4: If a match if found split follow the decomposition tree of the matched to split this block further.
Step 5: If there is no match, follow the algorithm 4.1 for decomposition.

Chapter 6

Conclusion And Future Work

From the two discussed box decomposition approaches mentioned in chapter 3 and 4, the Top Down Approach in chapter 4 takes a little longer time to execute than the region growing algorithm. Also the number boxes by the two algorithms are quite different. The region growing algorithm is much more sensitive to noise and missing data. So, the number of boxes is much more than algorithm 4.1. Also, algorithm 4.1 gives better result for missing data by avoiding creating holes in many cases. Table 6.1 shows the comparison. Using object recognition from repository models we can form a box decomposition from its partial model without creating holes. However, this technique totally dependent on how well the object recognition works. At present only our framework can only recognize parts which are very similar models in the repository. Also, we only follow decomposition of the matched model faithfully not modify it to suit current data.

In Future, we want to incorporate global features like VFH [22] and CVFH [21] along with aspect ratios for initial selection of the models from the data base. Also, we want use neighbourhood relations and not just point cloud matching for recognition. Also we want to add more primitives as cylinder and prisms which would give more visually better result.

Model Name	Algorithm 4.1 Time Taken(secs), No. Boxes	Algorithm 3.2 Time Taken(secs), No. Boxes
Fig. 4.5 (a)	1.8, 5	1.89, 19
Fig. 4.5 (c)	5.77, 5	3.52, 25
Fig. 4.7 (c)	16.02, 10	0.88, 35

Table 6.1: The above shows the comparison between Algorithm 4.1 and 3.2

References

[1] Olsson, Carl., Enqvist, Olof. Stable Structure from Motion for Unordered Image Collections. SCIA, Ystad, Sweden. Volume 6688, pages 524 - 535, 2011

[2] Triggs,B., McLauchlan, P., R.I, H. and Fitzgibbon, A. Bundle adjustment - a modern synthesis. In Vision Algorithms, pages 298–372, 1999.

[3] Chao-Hui Shen, Kang Chen, Shi-Min HU. Structure Recovery by Part Assembly. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia), Volume 31, pages 180:1-180:11, 2012.

[4] Ovsjanikov,M., Li ,W., Guibas, L., AND Mitra, N.J. Exploration of continuous variability in collections of 3D shapes. ACM Trans. Graph. Volume 30, pages 33:1-33:10, 2011.

[5] Kalogerakis, E., Hertzmann, A., AND Singh, K. Learning 3D mesh segmentation and labeling. ACM Trans. Graph. Volume 29, pages 102:1-102:12, 2010.

[6] Mitra, N.J., Guibas, L.J., AND Pauly, M. Partial and approximate symmetry detection for 3D geometry. ACM Trans. Graph. Volume 25, pages 560-568, 2006.

[7] Rother, C., Kolmogorov, V., AND Blake, A. Grab-Cut: interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. Volume 23, pages 309-314, 2004.

[8] Schnabel, R., Wahl, R., AND Klein, R. Efficient RANSAC for point-cloud shape detection. Computer Graphics Forum. Volume 28, pages 214-226, 2007.

[9] Gurari, E. Data structures: Chapter 19:Backtracking algorithms. Cis 680.

[10] Jain, A., Thormahlen, T., Ritschel, T., AND Seidel, H.P. Exploring shape variations by 3d-model decomposition and part-based recombination. Comp. Graph. Forum. Volume 31, 2012.

[11] Huebner, K., Ruthotto, S. and Kragic, D. Minimum volume bounding box decomposition for shape approximation in robot grasping. IEEE ICRA. pages 1628 - 1633, 2008.

[12] Chevalier, L., Jaillet, F. and Baskurt, A. Segmentation and Su-perquadric Modeling of 3D Objects. Journal of WSCG, 2003.

[13] Barequet, G. and Har-Peled, S. Efficiently Approximating the Minimum-Volume Bounding Box of a Point Set in Three Dimensions. Journal Of Algorithms, volume 38, pages 91 - 109, 2001.

[14] Testuz R., Schwartzburg Y., Pauly M. Automatic Generation of Constructable Brick Sculptures. Eurographics 2013 Short Paper.

[15] Jyun-Yuan Chen, Hung-Jui Lai, and Chao-Hung Lin. Point Cloud Modelling Using Algebraic Template. International Journal of Innovative Computing, Information and Control, volume 7, 2011.

[16] Ibele, T. Building Generator Version 1.7

[17] Tombari F., Salti S., Di Stefano L. Unique Signatures of Histograms for Local Surface Description, 11th European Conference on Computer Vision (ECCV), 2010.

[18] Lowe D. G., Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, volume 60, pages 91-110, 2004.

[19] Petrelli A., Di Stefano L., On the repeatability of the local reference frame for partial shape matching, 13th International Conference on Computer Vision (ICCV), 2011

[20] Tombari F., Di Stefano L., Object recognition in 3D scenes with occlusions and clutter by Hough voting, Fourth Pacific-Rim Symposium on Image and Video Technology, 2010.

[21] Aldoma A., Blodow N., Gossow D., Gedikli S., Rusu R.B, Vincze M. and Bradski G., CAD-Model Recognition and 6 DOF Pose Estimation International Conference On Computer Vision, 3D Representation and Recognition (3dRR11), 2011.

[22] Rusu R.B., Bradski G., Thibaux R., Hsu V. Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. International Conference on Intelligent Robots and Systems (IROS) Taipei, 2010.