# An Operable System for LoD3 Model Generation Using Multi-Source Data and User-Friendly Interactive Editing

## Final Report

# Final Report

# An Operable System for LoD3 Model Generation Using Multi-Source Data and User-Friendly Interactive Editing

Armin Gruen, Gerhard Schrotter, Simon Schubiger, Rongjun Qin, Biao Xiong, Changlin Xiao, Jiaqiang Li, Xiao Ling, Sidan Yao

Singapore ETH Centre – Future Cities Laboratory

1 Create Way

#06-01 CREATE Tower

Singapore 138602

TEL: +65 6601 4053

Submitted to

National Research Foundation Singapore

Virtual Singapore R&D Program - Towards Automatic Acquisition of 3D City Models for Virtual Singapore

# CONTENTS

# 1 Executive Summary

Photo-realistic 3D city models that represent the physical and functional state of the city are necessary components of the nation's digital infrastructure. It facilitates much easier governing, planning, simulation, measuring and prediction, and provides numerous potentials in both scientific and industry applications. LoD3 models contain building roof and façade geometry, as well as the functions of its different components (windows, doors, etc.). Generating accurate and standard 3D city models is a manually tedious, decisively rich and non-straightforward process, and the current practice of is LoD3 city modelling still a manually intensive process.
Given the high demand for city-scale model production in the virtual Singapore program, we aim to develop an operable workflow that could produce LoD3 with the lightest possible manual involvement. A multi-data approach is used by integrating different sources of data including oblique imagery, aerial images, airborne/mobile LiDAR, and UAV images, to produce high quality LoD3 models that meet the CityGML standards. The workflow consists of three necessary work packages (WP) that develop techniques in 1) Geometry modelling, 2) Semantic labelling and texture mapping and 3) Integration of procedural modelling and interactive geometric editing. WP1 develops novel image-based and LiDAR based roof topography and façade geometry modelling with automated and semi-automated methods. WP2 applies data fusion techniques with the latest machine learning methods to perform land-cover classification, façade element attribution and texture mapping. WP2 also develops a preliminary proof of concept in change detection. To ensure high fidelity of the resulting models, WP3 develops novel visualization-driven editing procedures that efficiently correct errors of the models and integrate the procedural modelling workflow into the 3D reconstruction of buildings with regular geometric patterns.
The **major objective** of this project is to work towards an operationally feasible approach to generate city-scale LoD3 models, and provide preliminary proof-of-concept on efficient model maintenance, to facilitate the broader mission of the Virtual Singapore program in developing Singapore as a more intelligent and smarter city.
In the course of this project we have developed a number of novel approaches and algorithms for data processing. We have demonstrated their functionality with real data from our NUS Campus test field. However, still needed is the integration of the software packages into a fully functional operational system and a robustification of some components.

## 1.1 Introduction

Over the years many fully automated approaches to building extraction have evolved, but only very few were designed to be semi-automated from the very beginning. Very often, procedures are declared automatic but require so much post-editing that their status as automatic methods becomes questionable.
Object extraction in general consists of three steps: detection, reconstruction (geometric modelling) and attribution (semantic modelling). This sequence even defines a processing strategy, but also represents a path towards increased complexity. At the detection level cues like colour and Digital Surface Model (DSM) data have proven to be particularly valuable. They are

first used to separate a man-made object from vegetation and other natural features; then to distinguish buildings from other man-made objects. Good success has been reported with isolated houses. Complex urban structures still widely resist this approach.
In reconstruction one encounters a great variety of methods depending on the type of building, level of required detail, number of images, kind of image cues and image primitives used, and utilized external and a priori information, level of automation and operator interference.
Recently, there has been a clear trend towards the use of multiple (more than two) images, colour cues, early transition to 3D processing, and geometrical constraints. Image cues may involve texture, colour, shadows, and reflection properties. Image primitives include points, double- and triple-legged vertices, linear elements, and homogeneous regions.

## 1.2 Approaches

There are basically two fundamental approaches which may lead to progress in automation: (a) advancement in image understanding and (b) multi-sensor, multi-data techniques. While (a) is a very hard problem to solve with a very long-term perspective, (b) can be realized more readily. We are combining both approaches in this project.
For this purpose we are looking into the problems of
+ Geometric modelling by image analysis and point cloud processing
+ Landcover classification and semantic modelling
+ Generic modelling and UI (User Interface and editing) issues
We target a system of LoD3 model generation with operability potential by integrating three necessary sub-components: 1) semi-automatic 3D roof and façade geometry and topology reconstruction, 2) façade elements semantic modelling and procedural modelling of facades and 3) interactive 3D editing based on polygon editing and procedural generation. Our proposed research combines the reality-based and procedural modelling, adopts semantic image understanding and interactive editing to accelerate the overall workflow of LoD3 model generation and enables various semantic attributes.

# 2 Project Objectives

We targeted design and realization of an operable software system for LoD3 model generation. Thereby we distinguish two starting points:

a) A LoD2 model is already given. The task is then to add 3D façade information to complete a LoD3 model
b) A Lod3 model has to be generated from scratch

The system should integrate the following 5 components:

1) Automatic/semi-automatic generation of buildings and topography and reconstruction from images and point clouds
2) Texture mapping
3) Land/object cover and façade classification and procedural modelling
4) Interactive 3D editing using RGBID (red, green, blue, intensity, depth) data
5) Change detection and change modelling

Due to lack of time topic 5 has not been treated extensively in this project.
The starting point of our approach requires raw data in the form of images, which in the overall process are turned into structured models. While imagery is a required input, additional data sets

such as point cloud data from Lidar scans are also included in the process in order to make good use of all available kind of data.
Today very often only a surface model, achieved through automated image matching, is presented as a 3D city model (see the 3D buildings of many cities under Google Earth). This technique, while reasonable fast in the production mode, has distinctive disadvantages. While it may be sufficient for low resolution visualization, it becomes problematic when the goal is to build up a geo-spatial information system (for example, Google does not allow the user to zoom-in arbitrarily since data inconsistency quickly becomes visible). Individual objects are not addressed in these data sets and semantics are missing. The applications of such data sets are very limited. Therefore, in this project, our research aims at improving the output of automated image matching by combining reality-based and procedural modelling, and by establishing an interactive, computer-assisted workflow.

# 3 Raw Data

As raw data we had aerial images from a 5-head camera system (Leica RCD 30), aerial Lidar (ALS) data and Lidar data from a terrestrial Mobile Mapping System (MLS). UAV images could be produced, but were not used.

- 36,600 vertical and oblique images, captured using a Leica RCD 30 5-head camera system
  Forward overlap 60%, sidelap of 25% with deviation range not exceeding +/-5%.
  8956 x 6708 pixels, GSD ca. 10 cm, 35 degree oblique
- MLS: Riegl VMX 450 with Ladybug 5 camera.
  Specs: 40pts/sqm at 70m range with speed of 60km/hr
- ALS: 0.25 mrad, 100-500 kHz
  Specs: Point density: 8.03-9.72 pts/sqm; Point spacing: 0.24-0.35 m
- Accuracy (georeferenced with GCPs):
  ALS planimetry 0.5m, height 0.15m
  MLS planimetry 0.3m, height 0.1m

# 4 Research Methodology

In general, the limitation of existing approaches is the lack of reliability and completeness in reconstruction. Techniques which may work with very low resolution data will fail at high resolution. Therefore, very often a zoom into the presented models will show their deficiencies. To overcome these limitations, we are using a Digital Surface Model (DSM), which may either be produced by image matching or through LiDAR-based point cloud generation (or both) as starting data set An approximation, from which individual objects are extracted with the help of images (aerial and terrestrial), LiDAR point cloud data (also aerial and terrestrial) and possible additional data sources like images from UAV flights.
The following R&D lines (a detailed overview can be found in Figure 4-1) will implement our approach, outlined as follows. For the structuring of the project we established 3 Workpackages (WPs) which treat the aforementioned functional components.

**WP1: Automatic/semi-automatic building and topography generation from images and point clouds**

(a) Semi-automated reconstruction of the roof landscape from aerial images. We use an advanced image matcher to determine a surface model. In an operator-guided procedure we measure all key points of a roof automatically by multi-image matching and we use a structuring approach similar to CC-Modeler to model each individual roof as a combination of planar faces. This approach can be adapted to varying resolutions.
(b) A similar approach is chosen for the 3D modelling of the facades. Very often, especially in modern high-rise buildings, facades have a highly repetitive structure. This fact is utilized by generating measurement macros of single façade elements (e.g. windows) and using a cut and paste approach to generate all the other elements of the facade. This also allows us to bridge occluded areas, when there is no information, neither from images nor from point clouds.
(c) Since in aerial LiDAR point clouds only limited facade information is available we use Lidar point clouds only from Mobile Mapping Systems (MMS) for façade modelling
(d) In a final step the operator will decide which result he will use, the image-based or the point cloud-based.

**WP 2: Semantic enrichment and texture mapping**
Assigning attributes to the components of the models is an important issue for LoD3 modelling. We employ image analysis and machine learning techniques to enhance the semantic content of the 3D models. In line with image analysis, we also implemented texture mapping to assign photorealistic appearance to the models. The following objectives have been implemented:
(a) Semi-automated land-cover image classification for object recognition. Given multi-sensory data, we propose a new joint feature representation to improve the land-cover classification accuracy to separate different objects, as a step to support LoD2 modelling.
(b) Semi-automated façade element identification. We will take the RGBD data corrected by WP3 to identify individual façade elements such as window, door etc. A façade element library of the test area will be built to support the generation of representative features and visual signatures of these elements.
(c) Given the I/O protocol of CityGML models from WP1, a view-independent texture mapping will also be implemented to assign photorealistic textures to the LoD3 models. The resulting texture mapping errors will be edited by WP3 with their interactive editing tools.

**WP3: Building procedural modelling and façade elements parameterization**
The main goal of this WP was to devise and realize a workflow that integrates semi-automatic extraction techniques with interactive tools, so that the overall modelling process becomes operable and streamlined, and allows to scale from buildings to blocks and eventually to towns and cities. This work can serve as a starting point to analyze the extraction methods resulting from WPs 1 and 2 and to identify their requirements and/or potential for integrating them into the computer-assisted modelling workflow.

**Interactive 3D editing using RGBID (red, green, blue, intensity, depth) data**
Procedural modelling methods for facades have scalability issues due to the low reliability of automatic image processing methods for complex façade types. In this WP we focused on improving these methods a) by streamlining the modelling process for particular output attributes (e.g. deducing specific material types or detailed depth structure for CFD simulations), and b) by revisiting recent developments in image recognition and inverse procedural modelling and developing them to a point where they become operable.

The three Workpackages do not operate separately from each other but are interconnected. Figure 4-1 and Figure 4-2 show the flow of processing routines and the connections between the Workpackages.

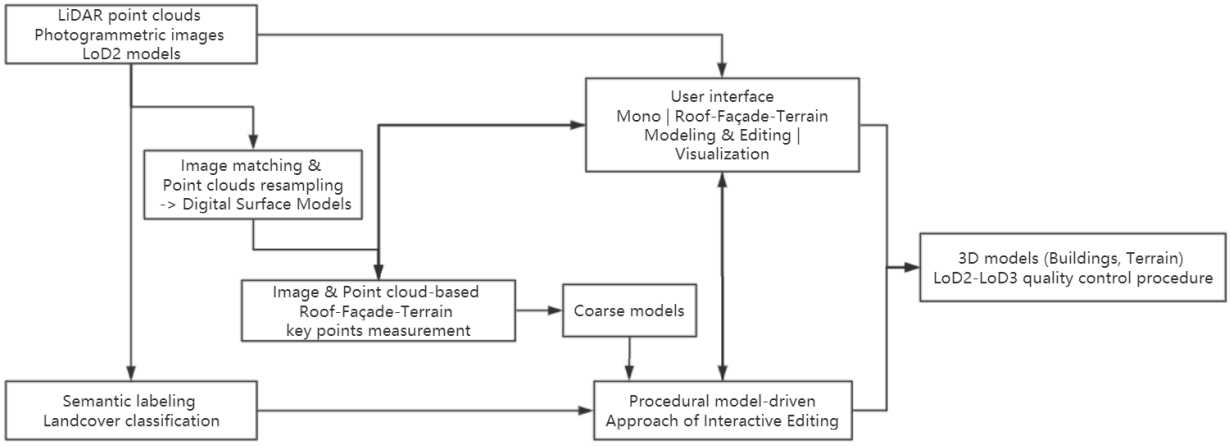



*Figure 4-1. Overall modelling workflow*

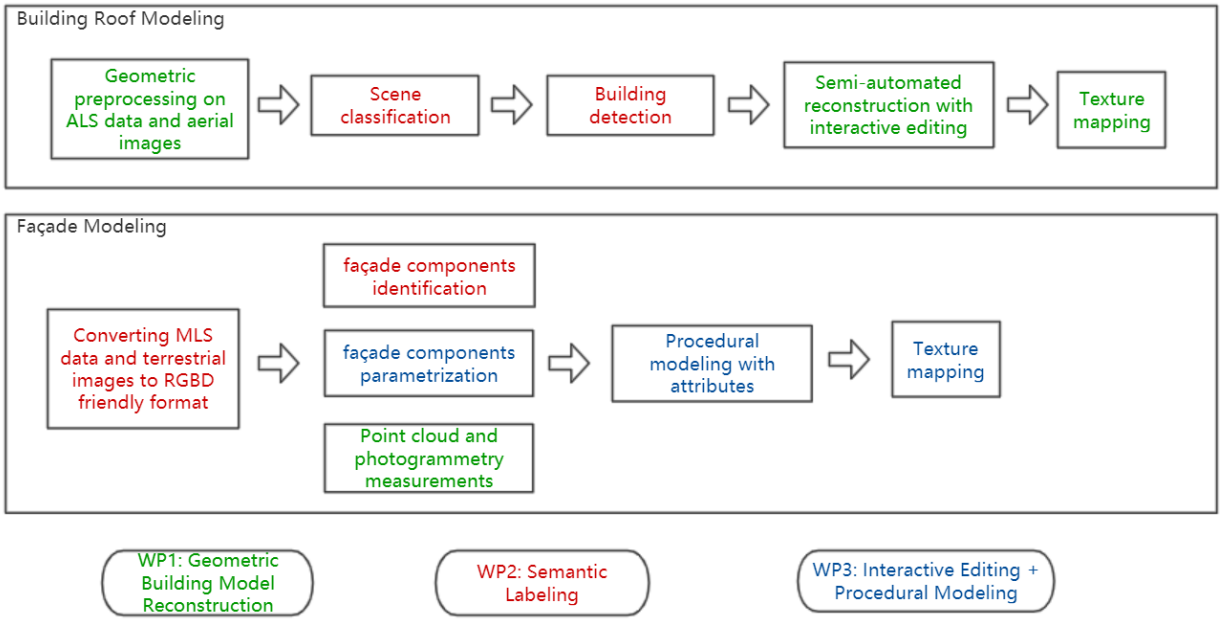



*Figure 4-2. Functionalities of work packages integrated into the overall workflow*

# 5 R&D Results

The following chapters will report about the results which have been obtained in this research project.
The individual Workpackages will report separately but it should be kept in mind that they are interconnected.

## 5.1 WP1: Automatic/Semi-automatic Building Model Reconstruction by Use of Images and Point Clouds

### 5.1.1 Accuracy Test of the Multi-Source Input Data

#### 5.1.1.1 Introduction

The multi-source input data for the Virtual Singapore project include 5-head camera images, airborne LiDAR point clouds, mobile LiDAR point clouds and the LoD2 building models from SLA. Since the quality of the orientations of these input data has great impact on our modelling work, we set up a test to check the georeferencing accuracy of the data. We selected the area of NUS campus, because this will be the pilot area for our Virtual Singapore project work. We firstly used 16 already existing Ground Control Points (GCPs) to check the accuracy of the image orientations, then check the georeferencing accuracy among the images, the LiDAR point clouds and the LoD2 building models. We also report about some problems in the LoD2 modelling results.

This accuracy test has three parts: 1) Accuracy test for the georeferencing of the 5-head images; 2) Georeferencing accuracy between 5-head images derived point clouds and LiDAR point clouds; 3) Georeferencing accuracy between 5-head images and the LoD2 building models from SLA.

#### 5.1.1.2 Accuracy Test for the Georeferencing of the 5-head Images

The quality control work herein includes two aspects: re-projection error statistics and forward intersection error statistics. The following two sections will discuss the details.

The GCPs for testing have a good distribution in the NUS campus area as shown in Figure 5-1.

*Figure 5-1. Distribution of the GCPs. The locations of the GCPs are marked with the green triangles and the names are in red. Area size ca. 2.2 sqkm*

**a) Re-projection Error Statistics**

All the GCPs are re-projected into the aerial images by applying the corresponding orientation parameters. The actual image coordinates of GCPs are manually measured. The measurement error is less than 1 pixel. The difference between re-projection image coordinates and the actual image coordinates is called re-projection error. The re-projection errors statistics of all GCPs are listed in Table 5-1 and Figure 5-2 shows the errors graphically.

*Table 5-1: Results of re-projection analysis*

| Camera type | Number of images involved | Number of GCPs involved | Number of observations | Re-projection errors (pixel) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Along image column | | | | Along image row | | | |
| | | | | AVERAGE | STDEV | RMSE | \|MAX\| | AVERAGE | STDEV | RMSE | \|MAX\| |
| Vertical | 33 | 15 | 87 | 5.4 | 1.0 | 5.5 | 8.5 | 0.3 | 0.9 | 1.0 | 2.1 |
| Forward | 31 | 12 | 51 | 5.7 | 1.9 | 6.0 | 7.6 | -0.1 | 1.8 | 1.8 | 4.8 |
| Backward | 18 | 10 | 26 | -3.4 | 1.9 | 3.9 | 5.1 | -0.8 | 2.1 | 2.3 | 4.9 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Left | 36 | 9 | 66 | 0.1 | 1.4 | 1.4 | 2.9 | -6.3 | 1.3 | 6.4 | 8.8 |
| Right | 23 | 9 | 32 | -0.1 | 0.9 | 0.9 | 1.5 | 7.4 | 0.9 | 7.5 | 9.0 |

STDEV: standard deviation
RMSE: Root Mean Squared Error

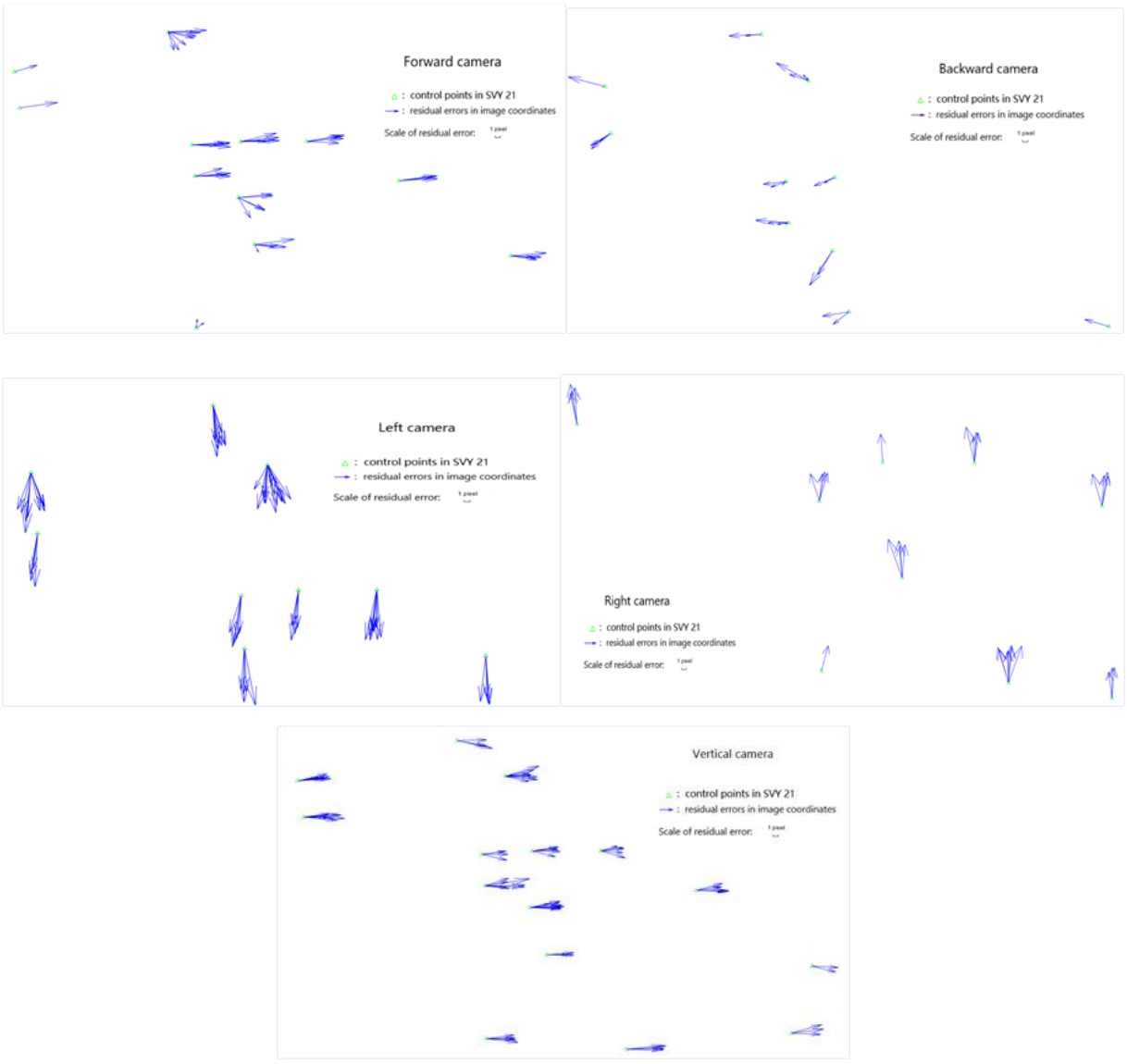



*Figure 5-2. Graphical presentation of re-projection errors in the 5-head cameras. The fact that the re-projection error vectors in the same camera are similar indicates that some local systematic errors exist in all cameras.*

Table 5-1 and Figure 5-2 show that the re-projection errors of the GCPs in the same camera are very similar both in size and direction. This indicates that there exist some local systematic errors, the sizes of which are between 2 and 6 pixels, depending on the camera.

**b) Forward Intersection Error Statistics**

All the actual image coordinates of each GCP from different images are forward intersected to obtain the world coordinates in SVY 21. The difference between the forward intersection coordinates and the actual world coordinates of the GCPs is called the forward intersection error. The forward intersection error statistics of all GCPs are listed in Table 5-2 and graphically represented in Figure 5-3.

*Table 5-2: Forward intersection error statistics*

| GCP ID | Number of images involved | Difference between the actual coordinates and the forward intersection coordinates (m) | | | Sigma0 of the forward intersection (pixel) | |
|---|---|---|---|---|---|---|
| | | Easting | Northing | Height | Along column | Along row |
| Point2 | 20 | 0.03 | -0.32 | -0.37 | 1.40 | 1.80 |
| Point3 | 22 | -0.01 | -0.17 | -0.48 | 2.24 | 2.00 |
| Point4 | 30 | -0.06 | -0.22 | -0.36 | 1.80 | 1.73 |
| Point5 | 22 | 0.04 | -0.45 | -0.07 | 1.35 | 1.05 |
| Point6 | 18 | -0.01 | -0.36 | -0.11 | 1.54 | 0.91 |
| Point7 | 23 | 0.04 | -0.36 | -0.34 | 1.08 | 2.99 |
| point9 | 15 | 0.02 | -0.38 | 0.05 | 1.17 | 1.36 |
| point10 | 19 | 0.01 | -0.52 | 0.14 | 0.75 | 0.48 |
| Point12 | 8 | -0.01 | -0.20 | 0.28 | 1.64 | 0.53 |
| Point14 | 8 | -0.06 | -0.36 | 0.03 | 0.53 | 0.53 |
| point15 | 11 | -0.01 | -0.40 | 0.42 | 0.74 | 0.81 |
| point16 | 3 | -0.01 | -0.41 | 0.15 | 0.26 | 0.48 |
| point17 | 3 | -0.02 | -0.24 | 0.66 | 0.23 | 0.06 |
| Addpoint2 | 18 | 0.02 | -0.38 | -0.16 | 1.08 | 1.72 |
| Addpoint3 | 25 | 0.01 | -0.43 | -0.01 | 1.46 | 0.91 |
| Addpoint4 | 17 | 0.06 | -0.35 | -0.28 | 1.46 | 1.28 |
| AVERAGE | | 0.00 | -0.35 | -0.03 | 1.17 | 1.17 |
| STDEV | | 0.03 | 0.10 | 0.31 | 0.56 | 0.75 |
| RMSE | | 0.03 | 0.36 | 0.30 | | |

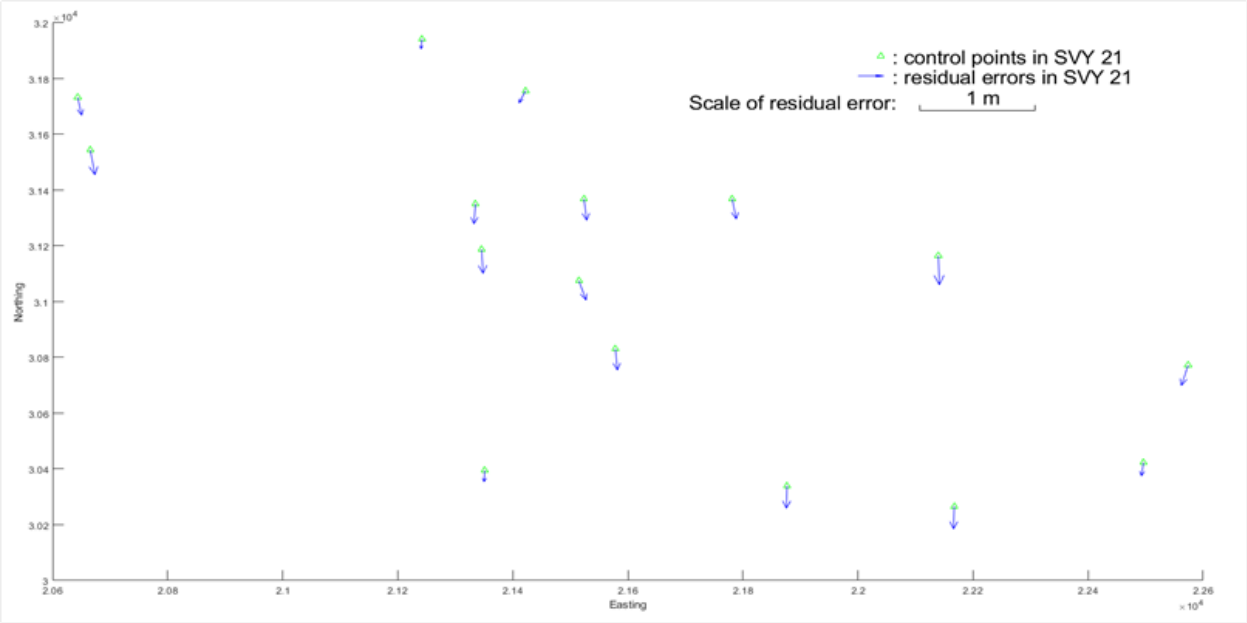



*Figure 5-3. Graphical representation of the forward intersection error statistics. All error vectors point towards the South.*

From Table 5-2 and Figure 5-3, we can see more clearly that there obviously exist systematic errors in the direction 'Northing', the size is about 0.35m. The accuracy of the relative orientation is around 1.5 pixels.

**c) New Triangulation with Pix4D Mapper**

The systematic error is quite significant in the absolute locations, while it is less obvious in the relative orientations. At this point it cannot be said where the systematic error comes from. For a thorough analysis one must know more about the bundle triangulation. One possibility could be that self-calibration parameters are not assigned independently to the individual cameras.
To avoid the undesirable consequences of the systematic errors to the follow-up computations, 306 5-head images covering the whole NUS campus have been selected as the input data to the commercial photogrammetry software Pix4D Mapper to redo the triangulation with the assistance of the GCPs. The geo-referencing accuracy after the bundle triangulation is about 0.029m, which meets the requirement of the modelling work. The orthomosaic and the corresponding sparse Digital Surface Model (DSM) produced by Pix4D Mapper from these images are shown in Figure 5-4.

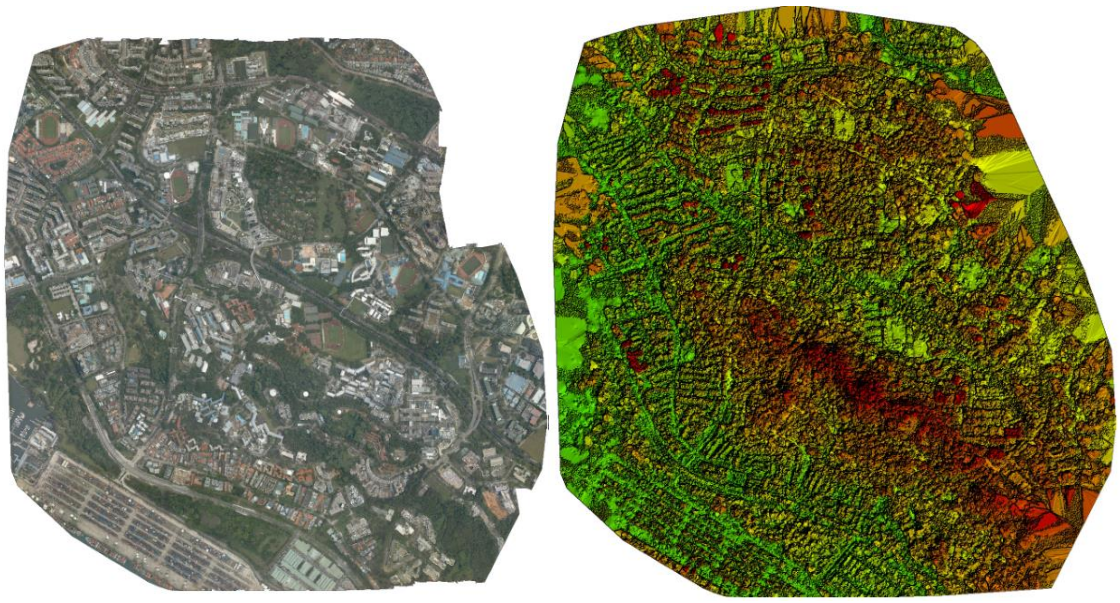
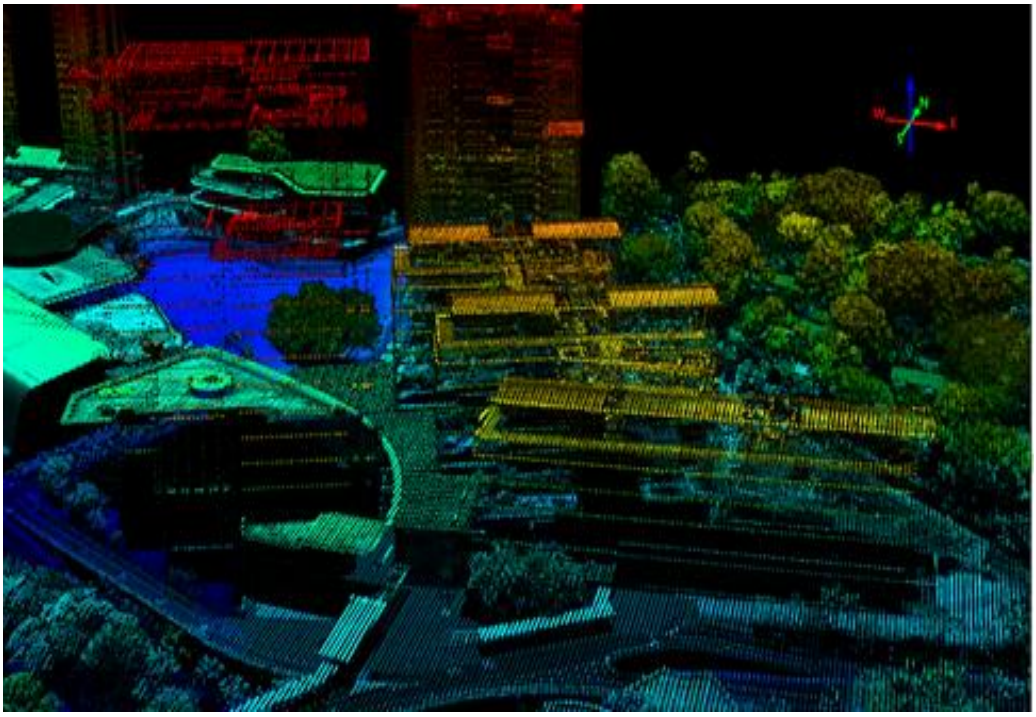


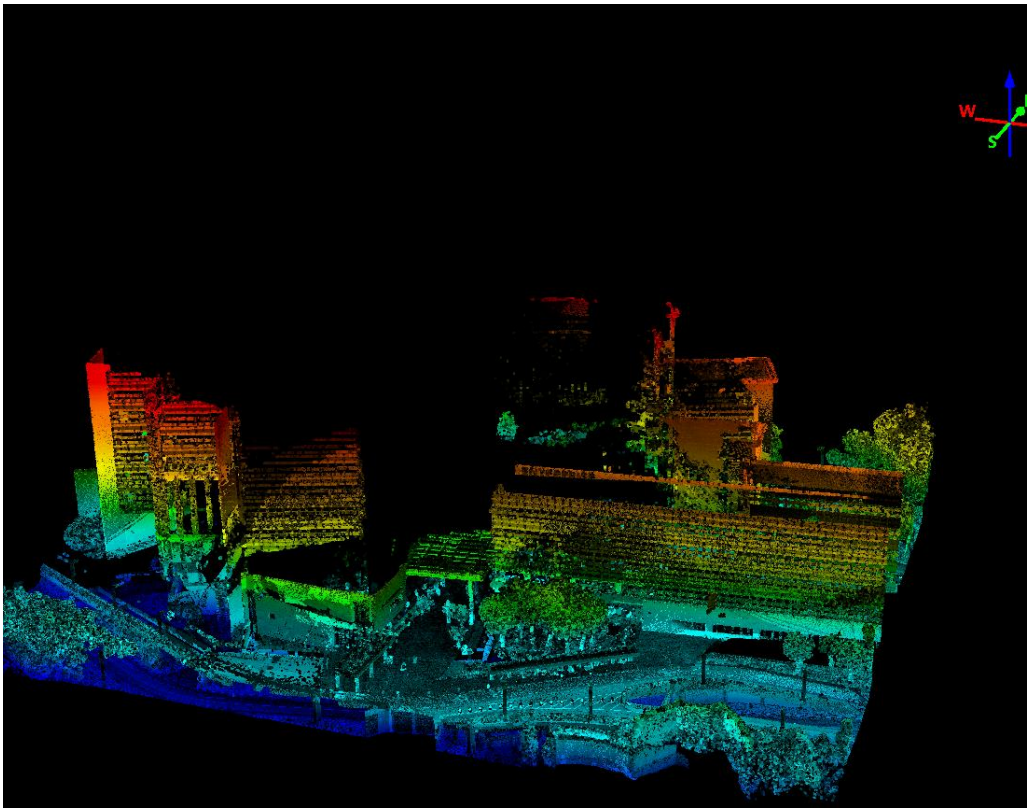
*Figure 5-4. Orthomosaic (left) and the corresponding sparse Digital Surface Model (right).*

#### *5.1.1.3* Georeferencing Accuracy Between the 5-head Images-Derived Point Clouds and LiDAR Point Clouds

The LiDAR data in the Virtual Singapore project have two sources: one is from the aerial laser scanning ( Figure 5-5 (a) ), the other is from the terrestrial mobile mapping system ( Figure 5-5 (b) ). From Figure 5-5, it is clear that the two sources of point clouds have different characteristics. The airborne LiDAR point cloud has more details on the roofs of the buildings and less, or even no information on the facade areas. On the contrary, we can see much more details, e.g. the openings, on the facades, but none about the roofs, or even the higher part of the buildings, from the mobile LiDAR point cloud.



(a) Airborne LiDAR Point Cloud




(b) Mobile LiDAR Point Cloud

*Figure 5-5. Illustration of CREATE tower from two sources of the LiDAR point clouds. The height is encoded by color.*

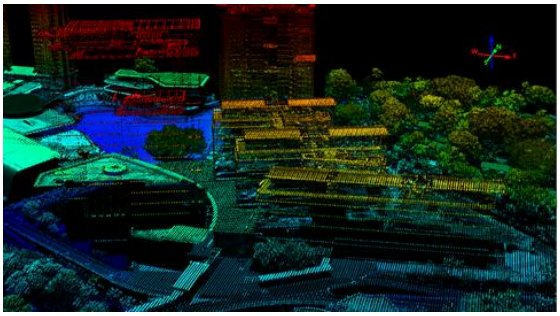
A dense point cloud has also been generated from aerial 5-head images by Pix4D Mapper to be compared with the LiDAR point clouds (see Figure 5-6 ). 306 vertical and oblique images were used and 197.3 Million points were generated.

*Figure 5-6. Dense point cloud of the NUS campus generated from 306 vertical and oblique images of the 5-head camera by Pix4D Mapper.*

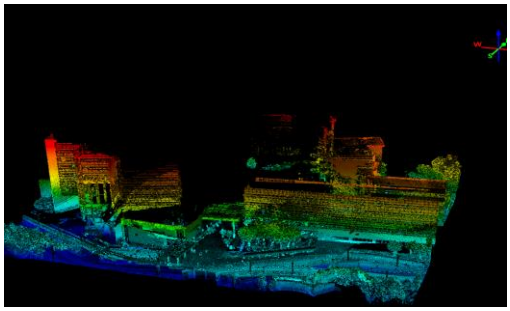
The quality control work herein includes two aspects: visual check and quantitative analysis. The following two sections will discuss the details.

**a) Visual Check**

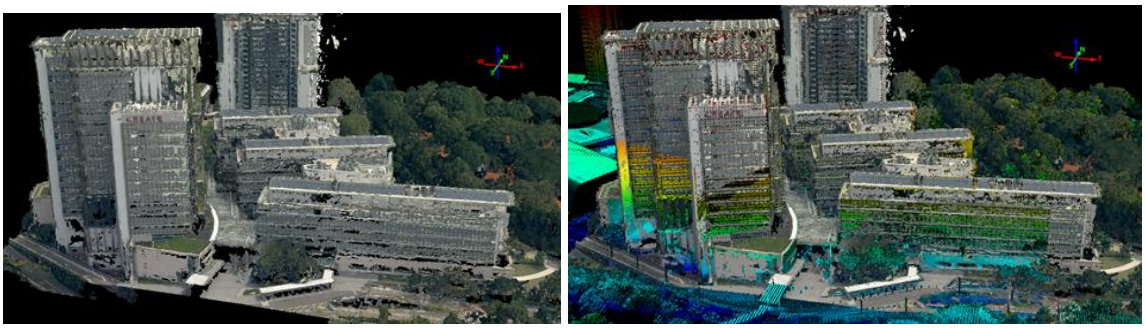
Two typical buildings are selected for the visual check: one is the CREATE tower, the other is the T-Lab in the Engineering school.



(a) Airborne LiDAR Point Cloud

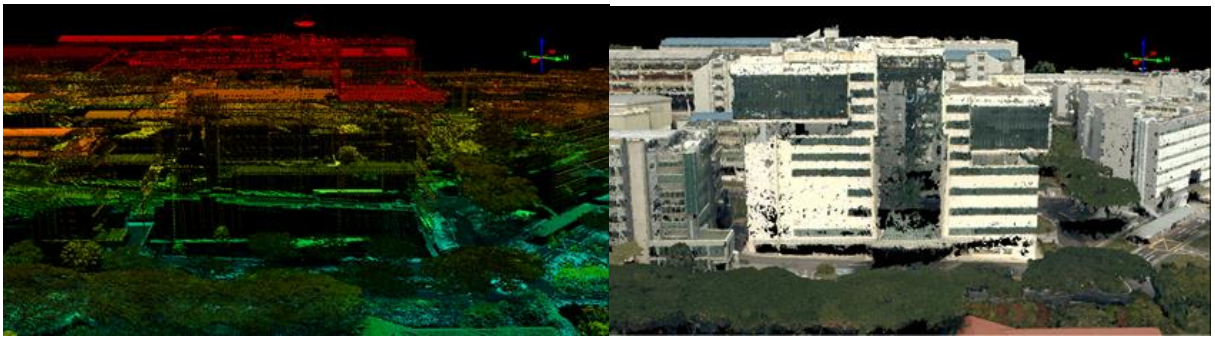


(b) Mobile LiDAR Point Cloud



(c) Point Cloud generated from 5-head images (d) Overlay of all three Point Clouds

*Figure 5-7. Overlay of multi-source point clouds of the CREATE tower.*



(a) Airborne LiDAR Point Cloud (b) Point Cloud from 5-head images

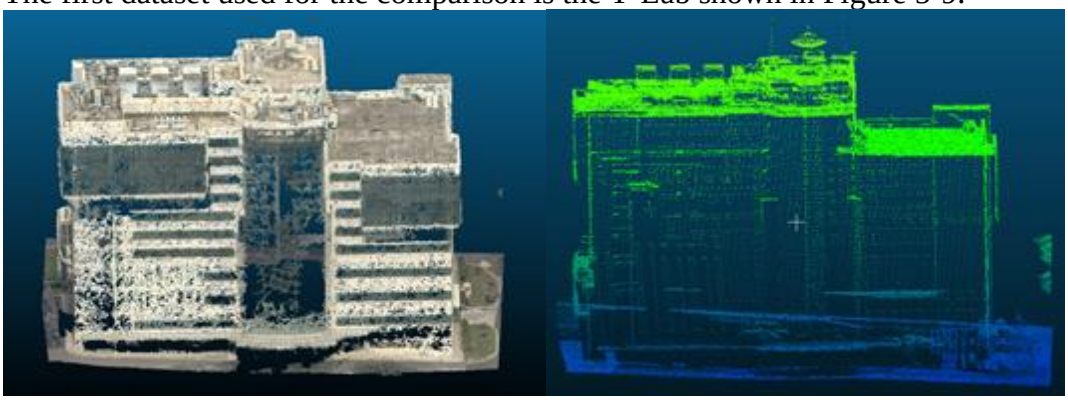
(c) Overlay of both point clouds

*Figure 5-8. Overlay of multi-source point clouds of the T-Lab.*

Figure 5-7. and Figure 5-8 show all three point clouds fit each other very well. There are no apparent mismatches. Thus, a more quantitative analysis is needed to get further details about the georeferencing accuracy.

**b) Quantitative Analysis**

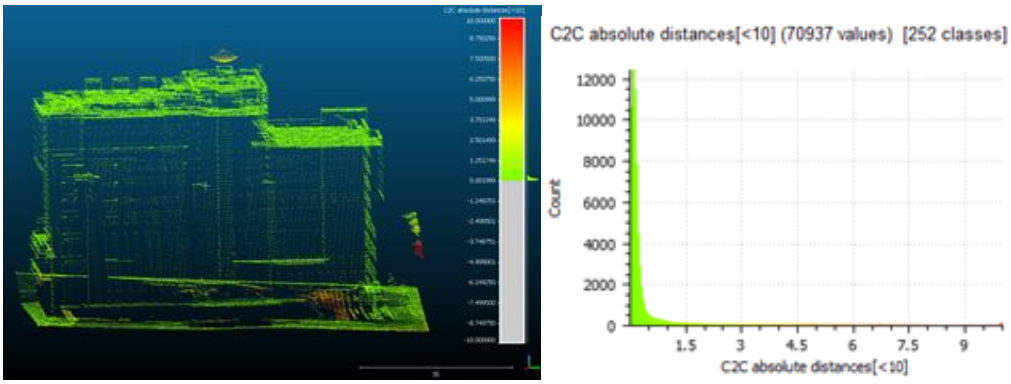
The common point cloud processing software CloudCompare is used in this section to obtain the absolute distances and X/Y/Z differences between the dense point clouds generated from 5-head images and the point clouds from the airborne LiDAR point clouds. When comparing two point clouds, we take one as 'reference' and the other as 'compared', the results of comparison, e.g. the difference in the X direction, will be used to color the point cloud 'compared'.
The first dataset used for the comparison is the T-Lab shown in Figure 5-9.



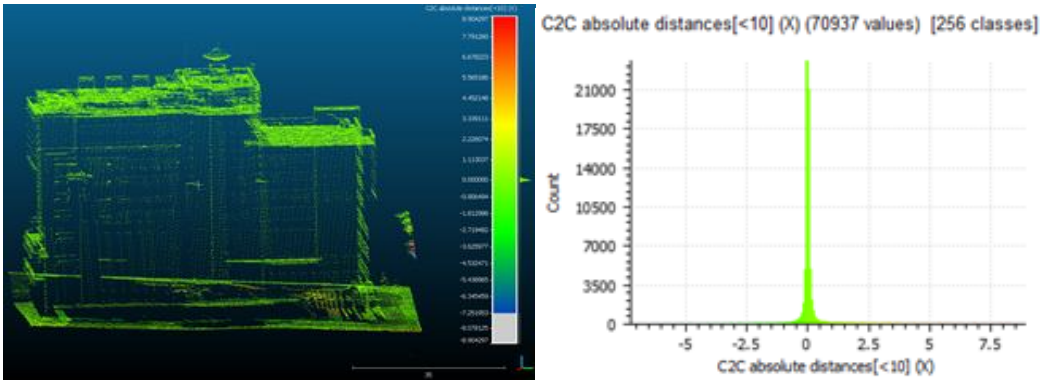
(a) Point Cloud from 5-head images (b) Airborne LiDAR Point Cloud

*Figure 5-9. First dataset for the quantitative analysis. (a) the total point number is 2,359,877; (b) the total point number is 70,937*

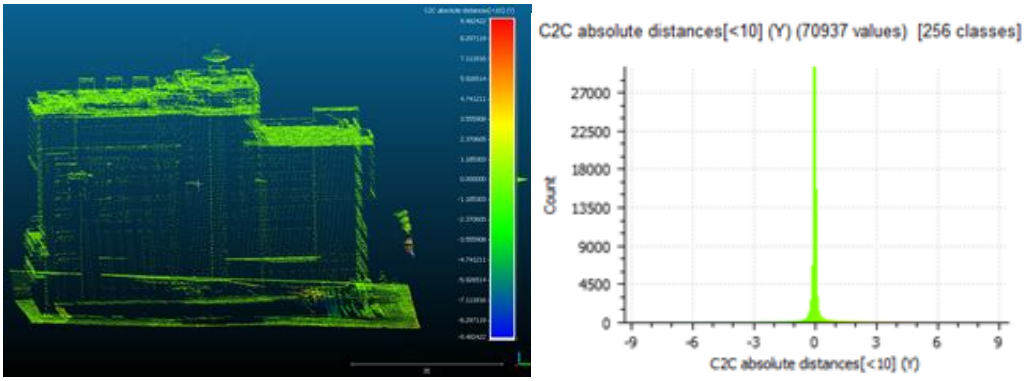
The point cloud from 5-head images is taken as 'reference' and the airborne LiDAR point cloud is 'compared'. The comparison result shows the mean distance between two point clouds is 0.502m while the standard deviation is 1.19m.



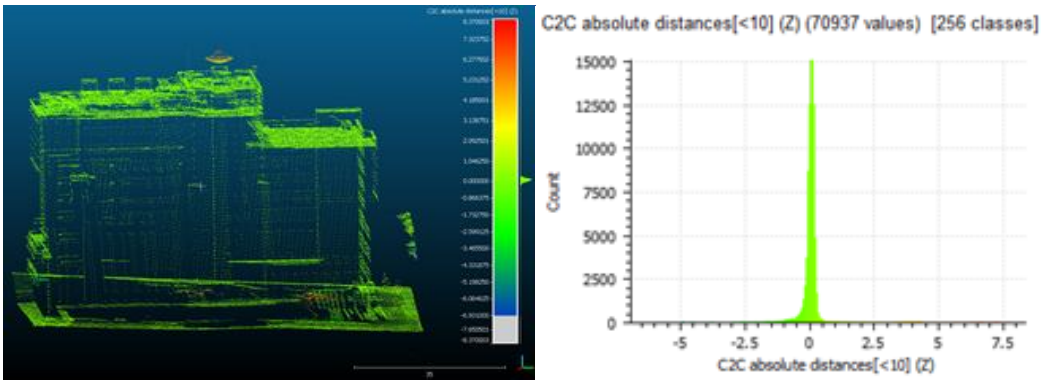

(a) Absolute distances (the mean distance is 0.502m)



(b) Differences in X (the mean difference is 0.13m)



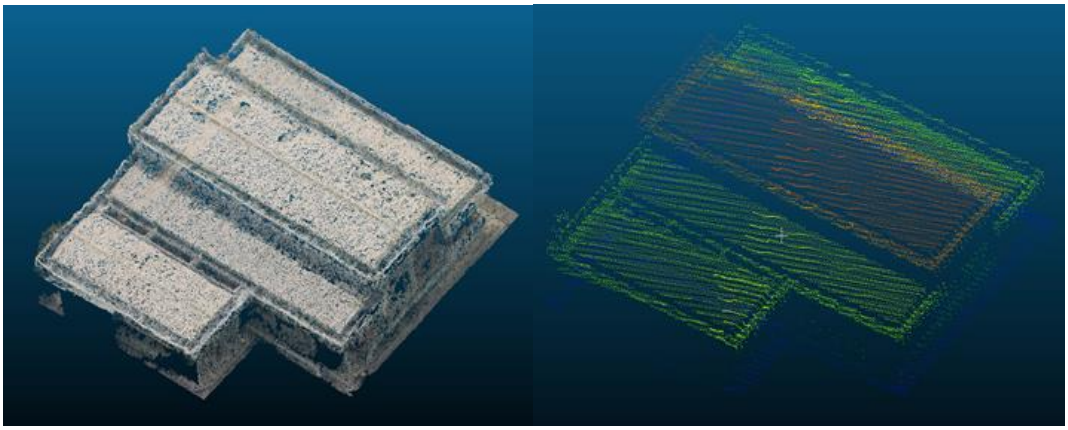

(c) Differences in Y (the mean difference is -0.08m)




(d) Differences in Z (the mean difference is -0.26m)

*Figure 5-10. Comparison results for the first dataset.*

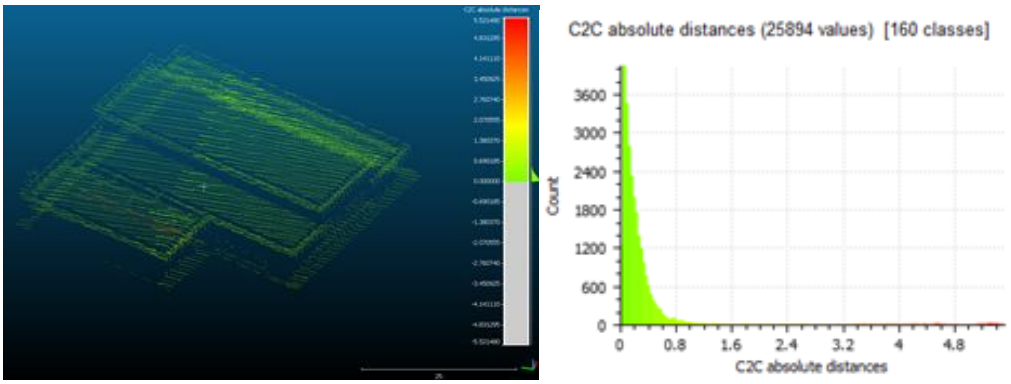
The second dataset is a low flat-roofed building, the corresponding point clouds are shown below.



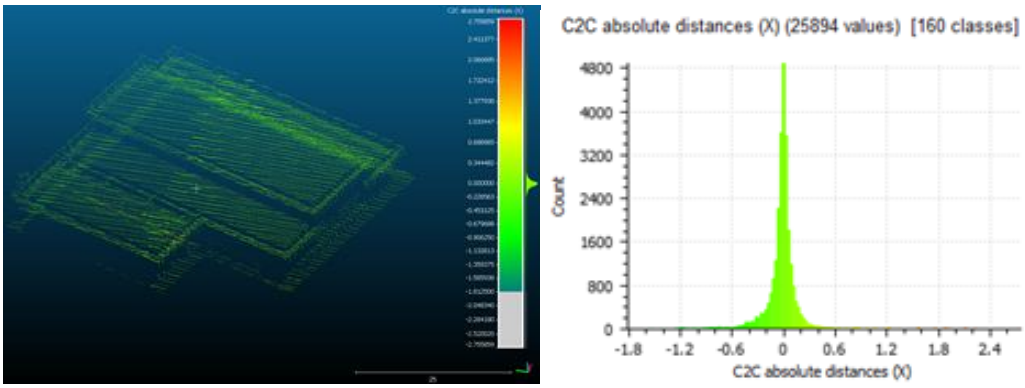
(b) Point Cloud from 5-head images (b) Airborne LiDAR Point Cloud

*Figure 5-11. Second dataset for the quantitative analysis. (a) the total point number is 588,244; (b) the total point number is 25,894*

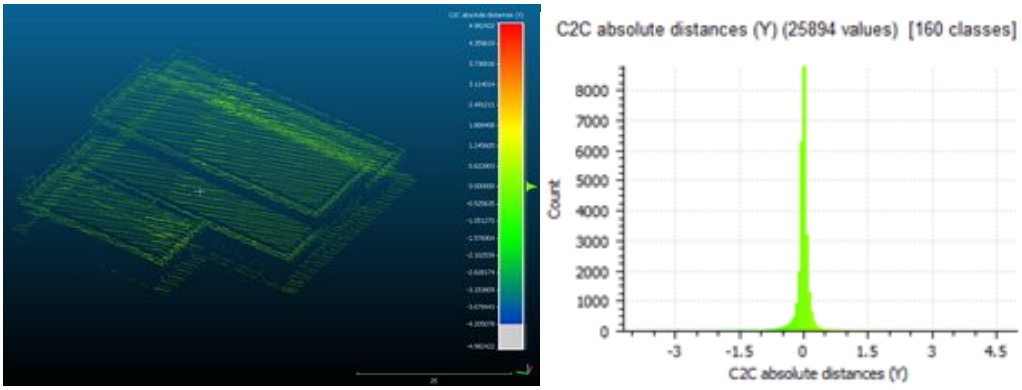
The comparison result shows the mean distance between two point clouds is 0.28m while the standard deviation is 0.54m.



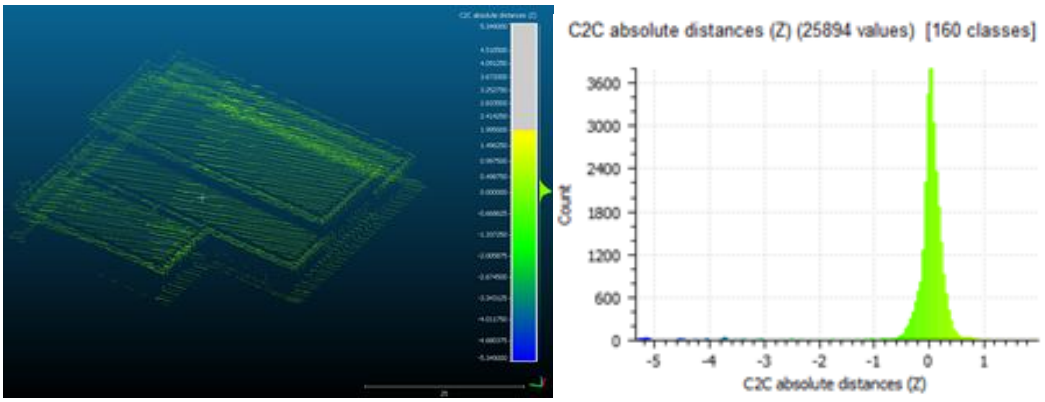

(a) Absolute distances (the mean distance is 0.28m)




(b) Differences in X (the mean difference is -0.01m)




(c) Differences in Y (the mean difference is 0.00m)




(d) Differences in Z (the mean difference is -0.14m)

*Figure 5-12. Comparison results for the second dataset.*

The quantitative analysis for two datasets is consistent with the accuracy specification of the airborne LiDAR data, 0.5m in planimetry and 0.15m in height.
It is difficult to further improve the accuracy since the low density of the LiDAR point cloud limits the ability to register it with GCPs. Thus, the LiDAR point cloud data is treated as assisted data for the images to provide the initial measurements and semantic information for detection and measurement of the facade elements.

#### 5.1.1.4 Georeferencing Accuracy Between 5-head Images and the LoD2 building models from SLA

The LoD2 models from SLA and the point cloud generated from 5-head images have been georeferenced. Figure 5-13 shows that they are well co-registered in the SVY21 coordinate system.

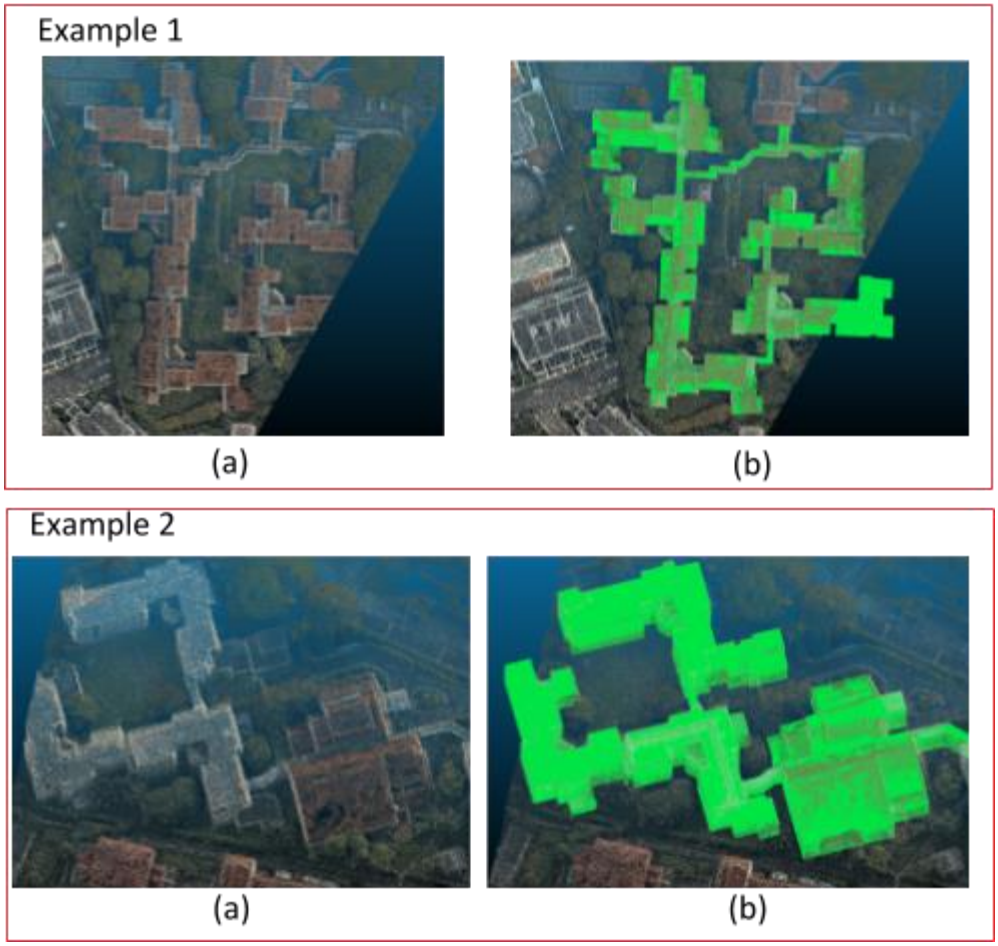



*Figure 5-13. Two examples of co-registration between the LoD2 from SLA and the point cloud generated from 5-head images for visual checking. (a) The point cloud and (b) the combination of point cloud and LoD2 model (green color without texture)*

A further analysis about the co-registration accuracy between these two source data is applied in the following steps: Firstly, the 3D coordinates of the polygons of the building elements (e.g. roofs and walls) are extracted from the LoD2 model. Secondly all polygons are reprojected to the images. Finally the image patches which contain some polygons are cropped out and saved. In the experiment, the roofs and the walls of a building are tested to check the co-registration accuracy.

The result in Figure 5-14 shows that the two datasets fit each other very well, the registration error of almost all polygons are less than 3 pixels. However, Figure 5-15 shows the limitations of the LoD2 models as well. There may be lots of blunders in the models, especially on the facades. Thus, the editing tool from WP3 is essential for processing the LoD3 models since it provides a user-friendly interface for operators to modify the errors.

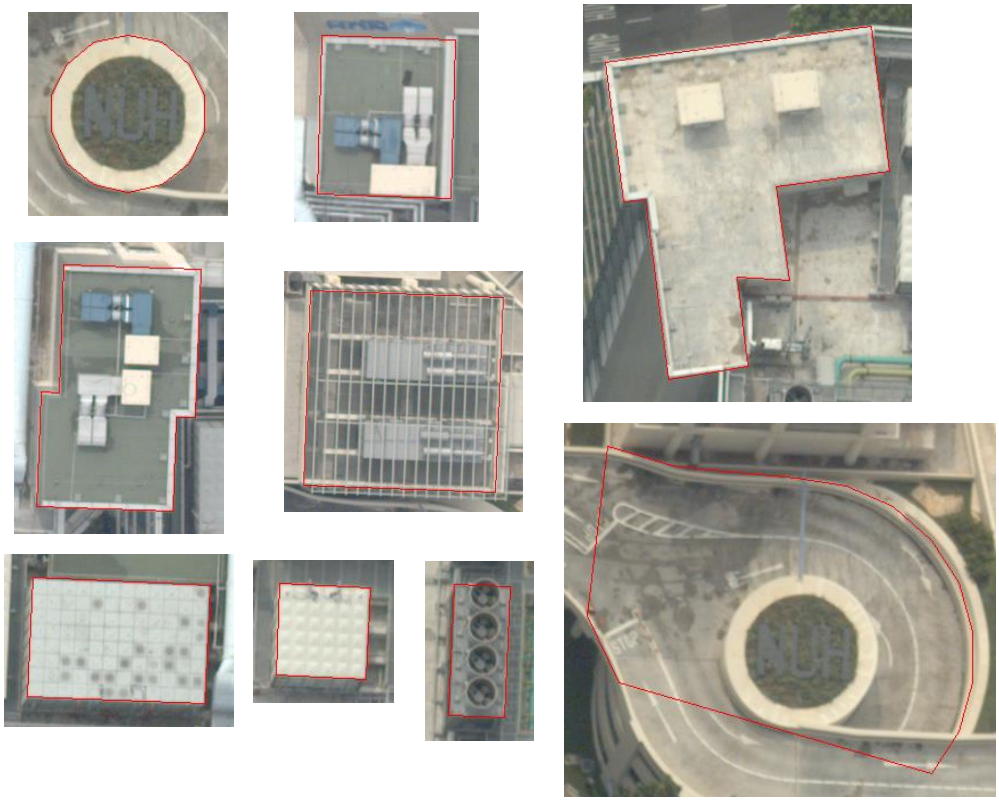


*Figure 5-14. Illustration of the co-registration between the vertical images and polygons from the LoD2 models (marked by the red polygons in the patches).*

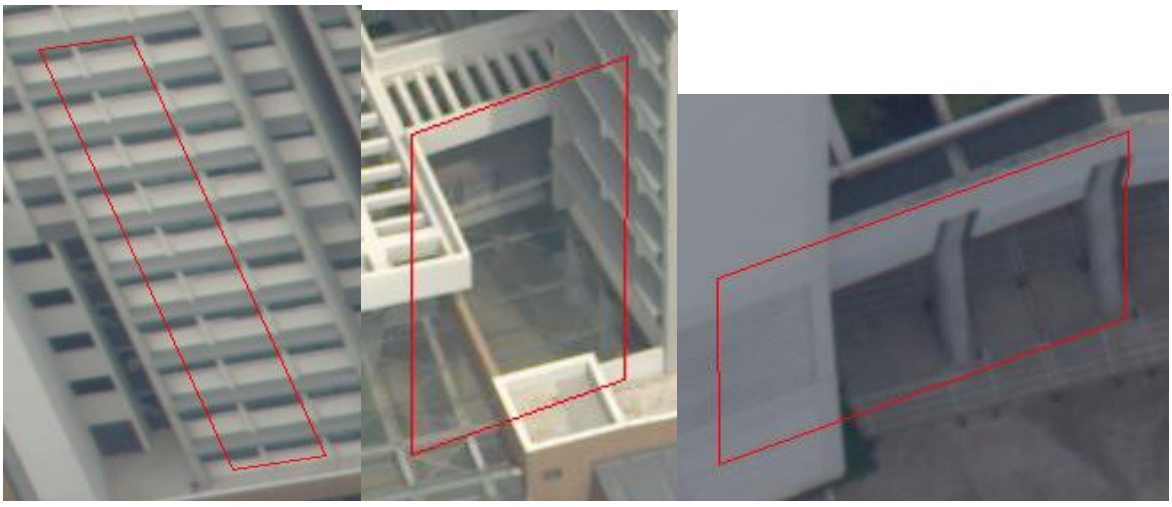


*Figure 5-15. Examples for blunders in the LoD2 models. The red polygons are the wall surfaces in the LoD2 models.*

### 5.1.2 Semi-automatic Point Measurement

#### 5.1.2.1 Introduction

Semi-automatic point measurement is the essential function in the semi-automatic building model reconstruction from images. It assists the operator to locate the positions of keypoints in

multiple views quickly and precisely by applying the multi-view matching method. The initial position of a keypoint in one image is from manual selection or building classification from WP2., The keypoint's location will be further refined by some keypoint detector, e.g. the Harris detector, and it will be re-projected to all images. However, it may not be visible in some images due to occlusions, thus a visibility check method has been developed to overcome the difficulty. Finally the multiple view matching method involving the least squares matching will obtain the accurate image coordinates in all images as well as the world coordinates of the keypoint. The overall workflow is shown in Figure 5-16. The details of the individual steps will be discussed in the following sections.

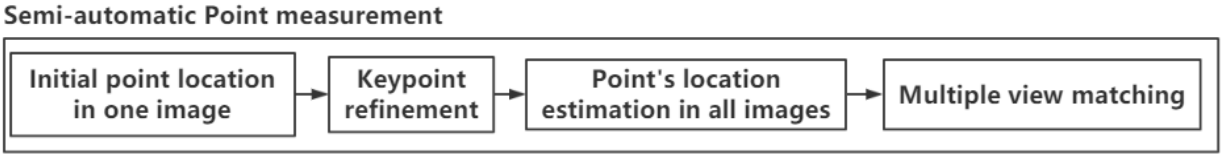



*Figure 5-16. The workflow of the semi-automatic point measurement.*

#### 5.1.2.2 Initial Location in One Image

The point's initial location may come from several sources:

- For roofs, since the LoD2 building models have fine roof structures, they can provide the initial locations of roof points;
- For façade elements, the initial location of their keypoints may be provided from a) façade opening detection from LiDAR point cloud in WP1, b) semantic labelling from façade image patches in WP2;
- If none of the above is applicable, the user has to select the initial point in an image via our interactive software.

#### 5.1.2.3 Keypoint Refinement

The initial point location usually does not define a keypoint well for some building elements, but it is very close to the correct keypoint location. Thus, some keypoint refinement algorithm (e.g. Harris point detection [1]) is applied to determine the keypoint's precise image coordinates. Keypoints should not be limited to only corners, but in some cases, such as when the object is a street lamp or a double-edge corner, the user may prefer the centre rather than the corner as the keypoint. To achieve this goal, we build image pyramids and apply the Harris detector not only to the original image level but also in the lower levels of the pyramid.
Two examples are listed as follows:(1) Double-edge corner: This is often seen along the building boundary.
(2) Blob, such as a street lamp. In this case users may use the centre to represent the blob instead of reconstructing its whole boundary. Thus, the Harris points detected from the upper level will be more in line with the needs of users.

*Figure 5-17. Harris detection for the double-edge corner. Nine candidate points from three levels of the image pyramid are detected and marked by different kinds of circles in the image. The size and the color of the circle imply which level the point comes from. Three red circles are Harris points from the original image level, three green ones are from the middle level and the three blue ones are from the top level.*

Figure 5-17 shows the keypoint detection result from applying the multilevel harris detector, both the centre and the outermost corner of the double-edge corner have been detected. Then, the user can choose whether the point near the centre or the outermost corner is taken as keypoint (Figure 5-18).

*Figure 5-18. Two Harris points from two different levels of the image pyramid. The Harris point shown in the left picture from the upper pyramid level can represent the centre of the double-edge corner while the right one from the original level is the outermost corner.*

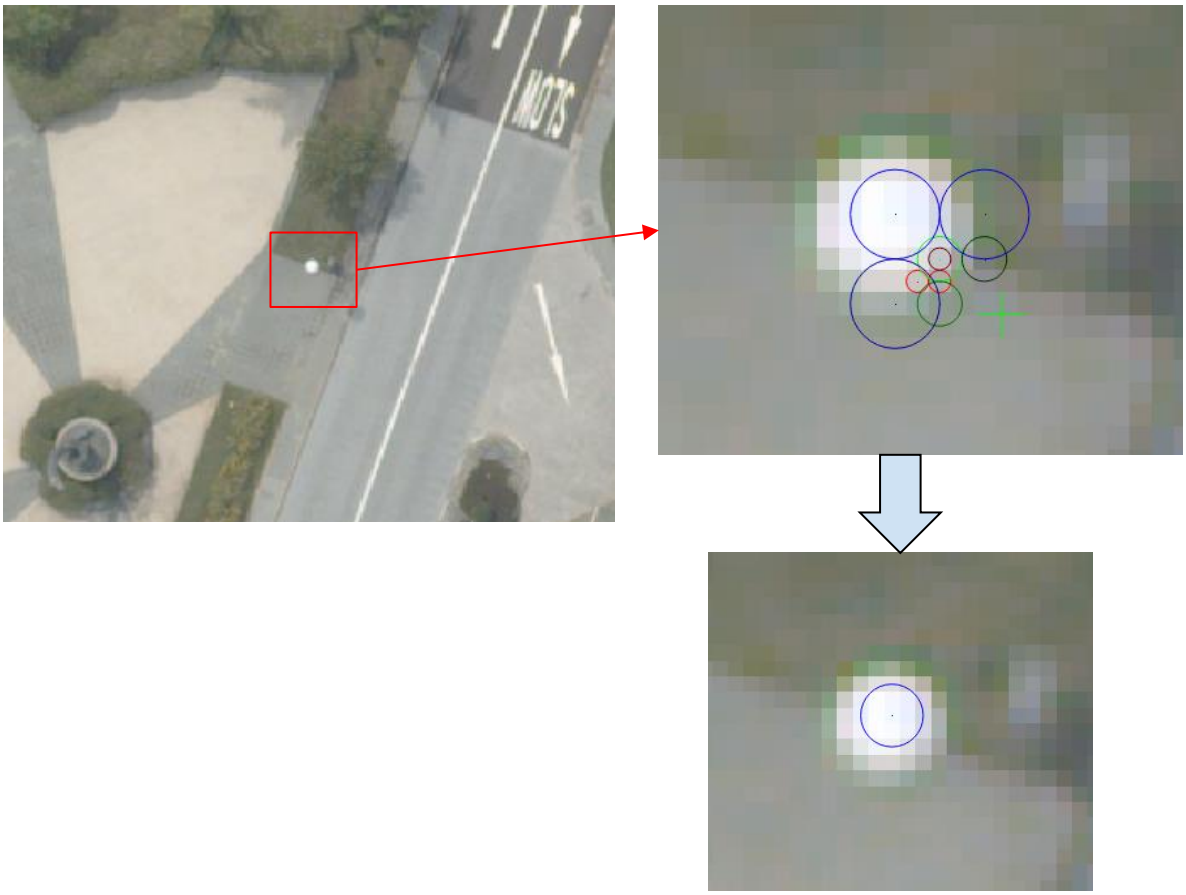


*Figure 5-19. Harris detection for the street lamp. In this case, Harris points detected from the original image are on the edge of the lamp (the red circles in the top right picture).*

Figure 5-19 shows the keypoint detection result from applying the multilevel Harris detector. Not only the outermost corners have been detected but also the centre of the street lamp. Compared to the corners, the user may more likely choose the centre (the blue circle in the bottom picture).

#### 5.1.2.4 Point's Location Estimation in All Images

Once the keypoint's location in an image is obtained, the point's corresponding approximate locations in all images can be estimated via forward-backward projection in which the corresponding DSM is employed to improve estimation accuracy.

*Figure 5-20. The estimated image coordinates of a keypoint (the roof's corner) in all images.*

However, the keypoint may not be visible in some images due to occlusions, as shown in Figure 5-21. This always reduces the smoothness of interaction and the stability of the algorithm. Thus we developed an occlusion detection method.
When the image coordinates of a point are determined, its 3D coordinates can be computed via forward intersection with the DSM. Thus, after a point is selected by the user in the major image and its corresponding coordinates in all other images are estimated by backward-forward projection, the point in each image will have its own 3D coordinates which should differ quite significantly from the one computed from the major image if there are occlusions. A distance threshold (1m) was set to detect the significant differences, that is if the distance between the 3D coordinates computed from one slave image and the 3D coordinates computed from the major image is bigger than 1m, the slave image will be marked as having a "occlusion".
The following example is used to show the effect. A keypoint is selected from the major image "IMG_094.tif" (Figure 5-21) and the occlusion detection result is shown in Figure 5-22.

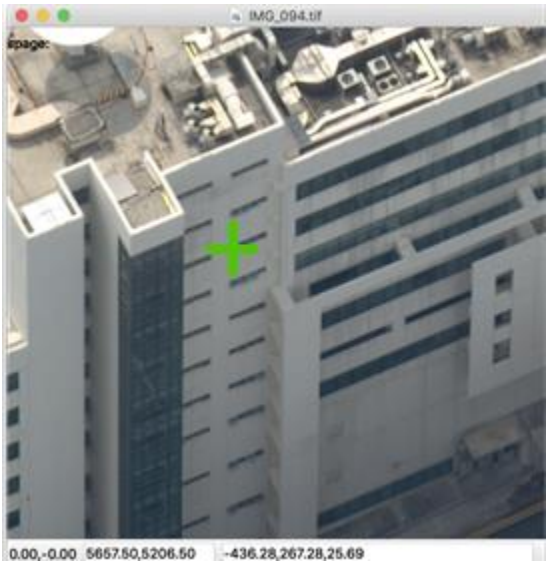



*Figure 5-21. Keypoint in the major image marked by green cross.*

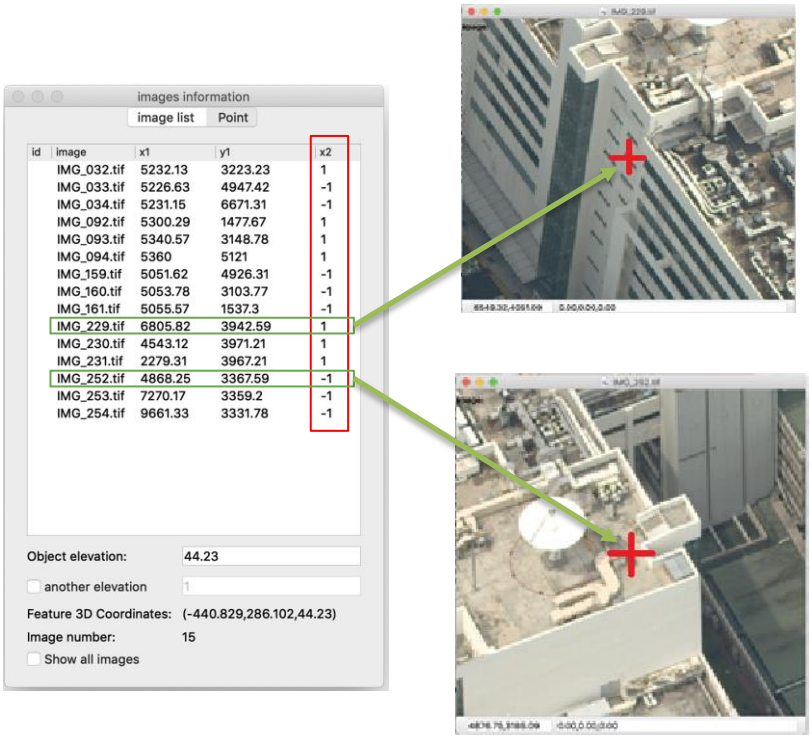



*Figure 5-22. Illustration of the occlusion detection. The column in the red rectangle is the occlusion detection result: -1 means there exists occlusion while 1 means no occlusion is detected. The two pictures on the right show that the results are correct.*

Figure 5-22 shows there are 15 images covering the selected keypoint while in 8 images (more than half of all) the corresponding points are not visible. These images can be removed before applying the multi-view matching method. While occlusions have negative effects not only on the multiple view matching process, but also on the facade texture generation, a further step is being developed to detect occlusions in the entire image.
The geometrical maps which record the 3D coordinates for every pixel in the 5-head images are generated as shown in the following figure.

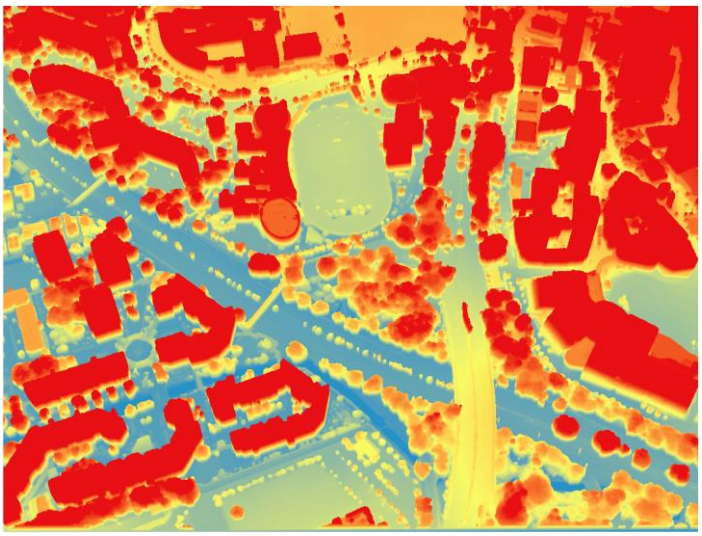


*Figure 5-23. Geometrical map corresponds to a forward-view image. The color is encoded by height.*

Based on the assumption that occlusion occurs in areas where the 3D coordinates suddenly change, an edge detector is employed to mark the fracture lines in the image. As shown in Figure 5-24, the boundaries of trees and buildings have been detected. More information about the boundary detection can be found in the WP2.

*Figure 5-24. Boundary detection. Trees (left) and buildings (right) are highlighted by detecting the fracture lines in the geometrical map.*

### 5.1.2.5 Multiple View Matching

A multiple view matching algorithm has been developed to accurately obtain the point's location both in image space and object space. This algorithm has two steps: one is epipolar-line based multi-view matching to provide the positions with reasonable accuracy in all images for keypoints and the other is geometrically constrained multi-view least squares matching to refine the positions with higher accuracy.

Two principles are applied in the epipolar-line based multi-view matching:

- The location of the keypoint detected in the master image is fixed through the matching processing
- The epipolar geometry is applied as hard constraint, that is assuming the geometric difference between any two image patches will be rectified according to it.

The epipolar-line based multi-view matching has three steps:

1) Image epipolar rectification for the master image and each slave image according to the epipolar geometry. As the orientation parameters of all images are obtained by bundle triangulation (here: the Pix4D Mapper software), the epipolar geometry between the master image and each slave image can be calculated, according to which each image pair (the master image and one slave image) are rectified [2]. In this step there are two parameters required, one is the height range of the keypoint in the 3D space which determines the length of the epipolar line. This can be automatically determined if there is a DSM. Another is the maximum possible error of the keypoint related to the epipolar line in the experiments. These two are set to 20 and 10 pixels respectively.
2) Template matching for each epipolar image pair. When the images have been rectified, the master epipolar image is taken as template image while the corresponding slave epipolar image is taken as source image in which we expect to find a match to the template image. The match metric map will be computed for each epipolar image pair using the comparison method Normalized Cross Correlation (NCC) (Figure 5-25). There are two algorithmic parameters in this step, one is the template image size, another is the comparison method, and they are set to 25 and NCC respectively.
3) Point location optimization: We assume that there are a total of $n$ match metric maps which are denoted by $M_i(x, y)$ where $i = 1,2,\dots,n$. In this case, the x coordinate of the match metric map is along the epipolar line and can be determined by the keypoint's height in 3D space. The y coordinate measures whether the corresponding point in the slave image is consistent with the epipolar geometry. So the cost function for searching the optimum point location is defined as follows:

$$e(h, y1, y2, \dots, yn) = \sum_{hmin}^{hmax} M_i(h, yi)$$

Where $h$ is the point's height, $y1, y2, \dots, yn$ are the y coordinates in the corresponding metric map, $hmin\ and\ hmax$ are the height range mentioned in Step 1. We will get the optimum point location when this cost function reaches its maximum value. To solve this function, each metric map $M_i(x, y)$ is compressed to one-dimension array $A_i(x)$ by taking the maximum value along the x direction for all columns, then summing all $A_i(x)$ and finding the maximum and the corresponding x, finally the keypoint's height will be calculated and the keypoint's corresponding image coordinates will be obtained. There is only one algorithmic parameter in this step, the match metric threshold (0.4 in the experiment) which is used to judge whether the matching between the template image and a source image is successful.

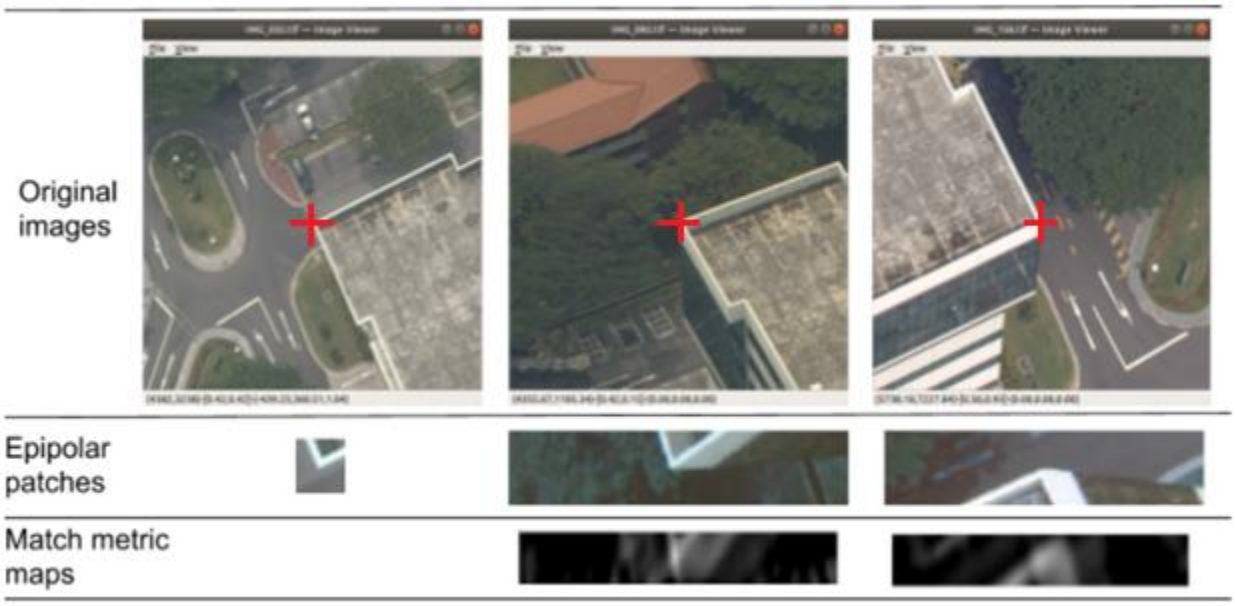



*Figure 5-25. Illustration of the match metric maps. The first column is the master image, the middle column is one slave image from forward view, its epipolar image and its match metric map; The third column is another slave image from backward view, its epipolar image and its match metric map. In the match metric maps, the brighter the pixel, the higher the correlation.*

The following is an example. A keypoint on the edge of a street lamp is detected to test the algorithm (Figure 5-26). Since the street lamp has not been modelled in the DSM, the estimated point locations in other views are far away from their true locations (Figure 5-27).


*Figure 5-26. Street lamp in the top view, the green cross is the keypoint detected by the Harris detector.*

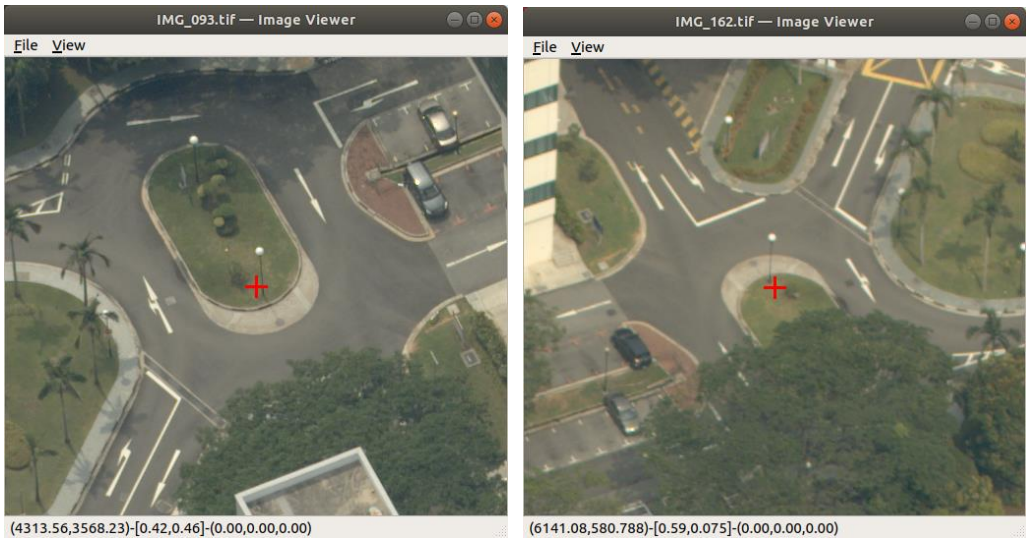



*Figure 5-27. Estimated locations of the keypoint (marked by the red cross) in a forward view (left) and a backward view (right). Since the street lamp does not exist in the DSM, the estimated locations are far away from their actual locations.*

After the epipolar-line based multi-view matching is applied, the estimated point locations have been corrected to their actual locations (Figure 5-28). This proves the algorithm really plays its role.

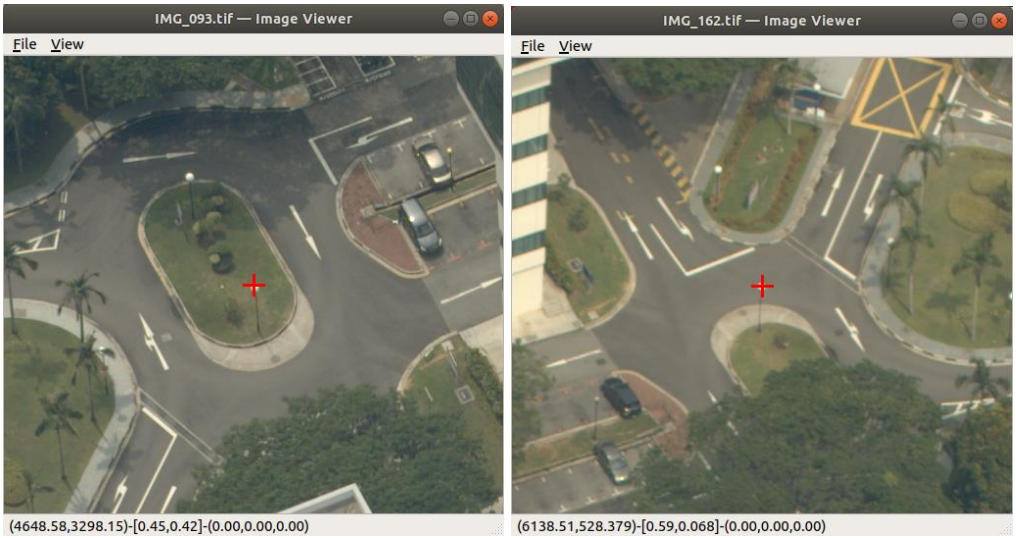



*Figure 5-28. Point locations after applying the epipolar-line based multi-view matching.*

Geometrically constrained multi-view least squares matching considers the geometrical difference as variables and solve them in the iterations under the epipolar line constraint. The common transformation model to describe the geometrical difference is the affine transformation which considers rotation, two scales, sheer and two offsets between two images [3, 4].
The following is the matching example for the corner of a window on a façade.

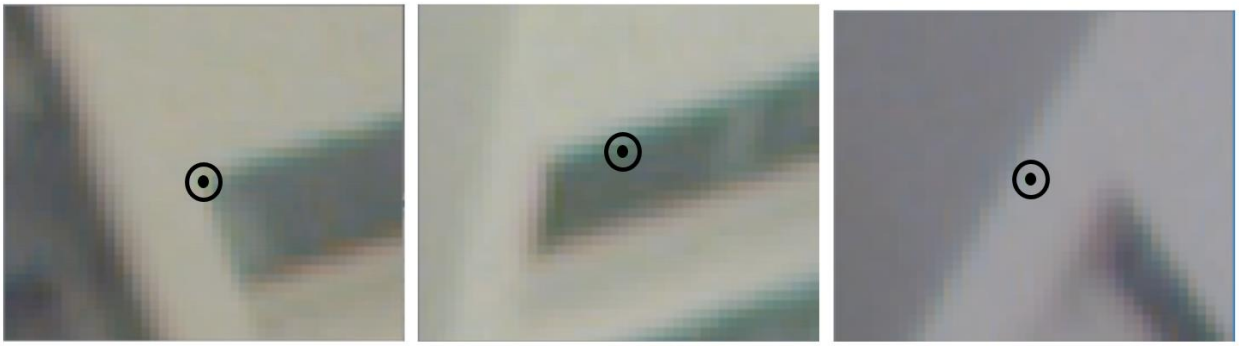


*Figure 5-29. Initial locations of match points. Left: master image with the keypoint marked by the black circle from the left view. Middle: slave image from the left view and the black circle is the initial location of the keypoint. Right: slave image from a backward view and the black circle is the initial location of the keypoint.*

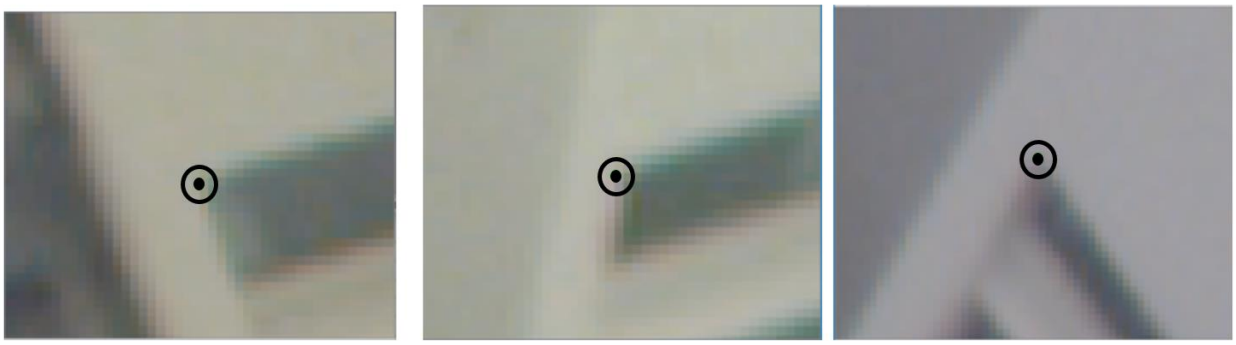


*Figure 5-30. Final matching result after a number of LSM iterations.*

One corner of a window in facade is selected as keypoint from a left-view image and totally 32 images contain it (see Figure 5-29). The initial location of the keypoint in every slave image is about 7 pixels off its optimum location. Figure 5-31 shows how the transformed slave images become more and more like the template image patch gradually in the iterations, which implies that the transformation parameters getting more and more accurate in the iterations and finally get close enough to the actual values. Figure 5-30 is the final matching result which shows all locations of the keypoint are accurate after the LSM is applied.

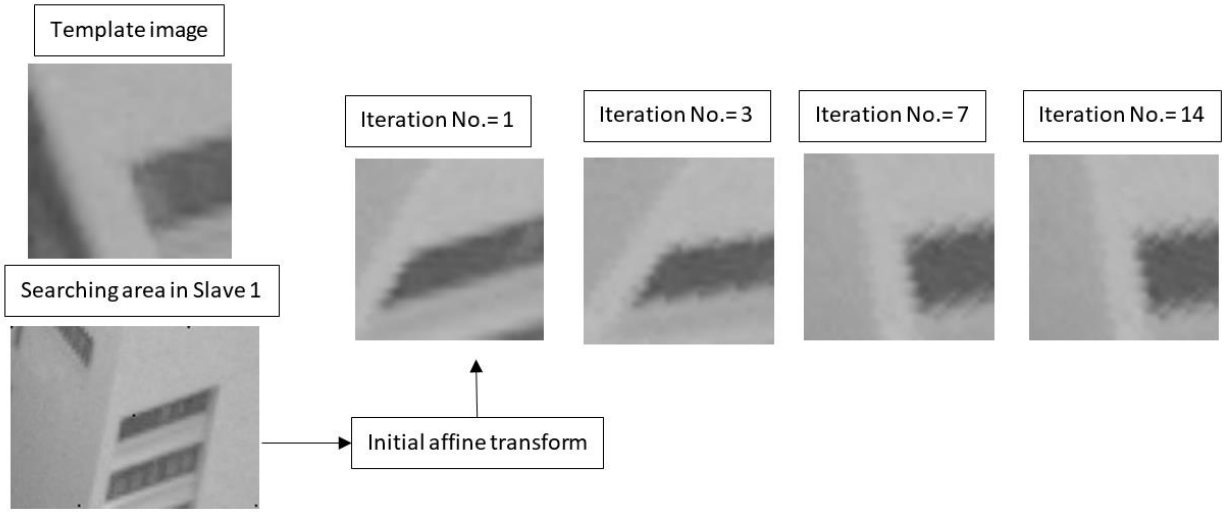



*Figure 5-31. Illustration of iterations in the least squares matching for slave image 1.*

## 5.1.3 **Measurement Macros Based Reconstruction**

### 5.1.3.1 Introduction

The primary objective of semi-automated geometric building model reconstruction is to reconstruct LoD3 building models from 5-head images and LoD2 building models if they do not exist.

We mainly consider three principles when designing the workflow: 1) minimize the amount of human interaction; 2) maximize the location accuracy of vertices of the building models, 3) optimize the reliability of reconstruction. Two concepts are introduced throughout the workflow to achieve these goals: 1) develop measurement macros; 2) apply least squares matching.

We will introduce our workflow and describe the details in the next sections.

### 5.1.3.2 Workflow

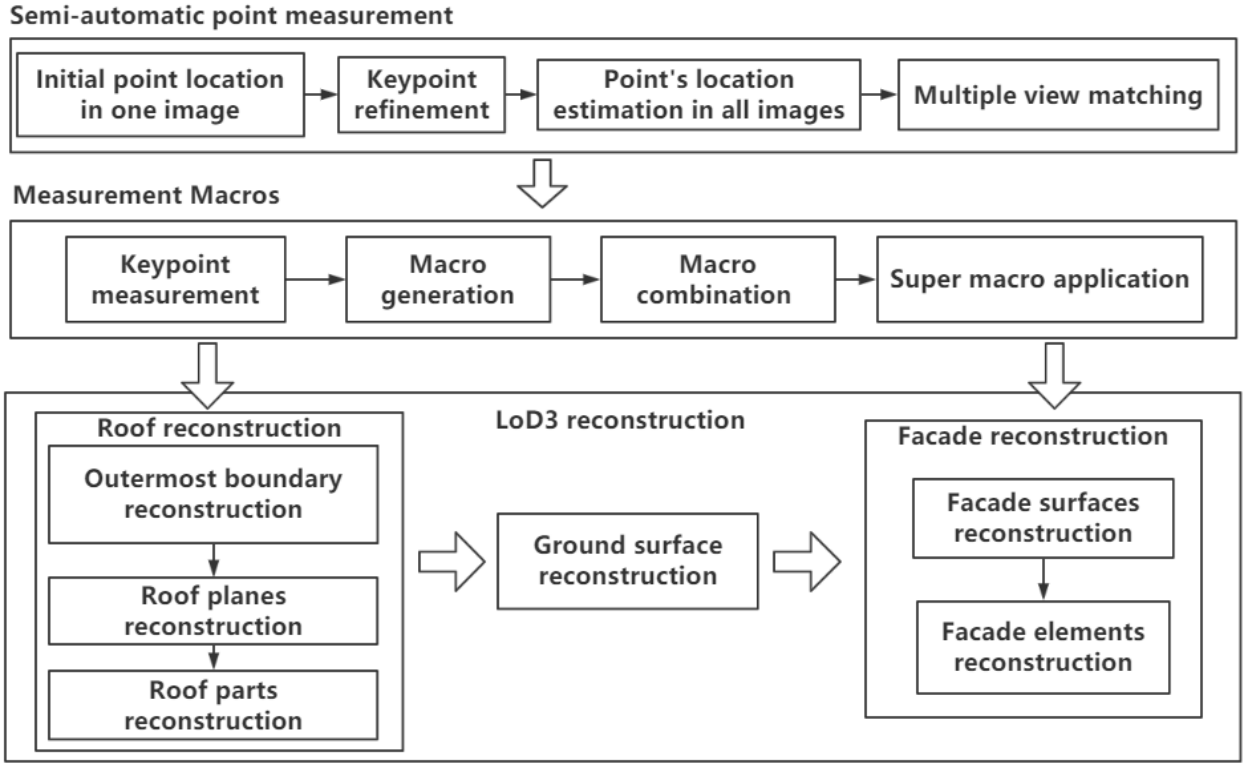



*Figure 5-32. Workflow of semi-automated geometric building model reconstruction*

There are three major procedures in the LoD3 reconstruction (Figure 5-32):

1) Roof reconstruction. This procedure is to reconstruct the roof's boundary, roof planes and roof parts above the roof planes step by step.
2) Ground plane reconstruction: we reconstruct a ground surface from filtering buildings trees and other unwanted objects out from the DSM.
3) Facade reconstruction. This procedure is to reconstruct the main facade surfaces, especially for the case where the roofs and the façade surfaces don't share boundaries, and façade elements step by step. Here we concentrate on reconstructing three typical façade elements: windows, doors and balconies.

The basic module "semi-automatic point measurement" is to accurately measure the point's location both in image space and 3D object space, and has been discussed in the previous section. It is applied to all procedures throughout the LoD3 reconstruction. Another basic module "measurement macros" is to efficiently reconstruct building elements via "copy and paste" in two procedures "roof reconstruction" and "façade reconstruction".

### 5.1.3.3 Roof Reconstruction

**a) Outermost Boundary Reconstruction**

There are three reasons for reconstructing the outermost boundary in the first step:

1) The roof's outermost boundary is easy to be modelled since it is usually a polygon.

2) The modelled boundary is useful for finding all images covering the building and is a constraint in the next steps. For example, the modelled roof parts outside the boundary will be marked as potential error to be checked.
3) Every line segment of the boundary corresponds to a facade surface in many cases This will be useful for the later procedure "facade reconstruction".

Two types of measurement macros are provided for users to model the boundary: 1) straight line; 2) arc. Almost all roof boundaries can be decomposed into these two shapes.
In this step, users choose a certain measurement macro "straight line" or "arc" depending on the roof's shape, then select key points successively along the boundary in an image (Figure 5-33). Finally, the accurate 3D location of the roof's boundary will be automatically obtained by the point measurement module.

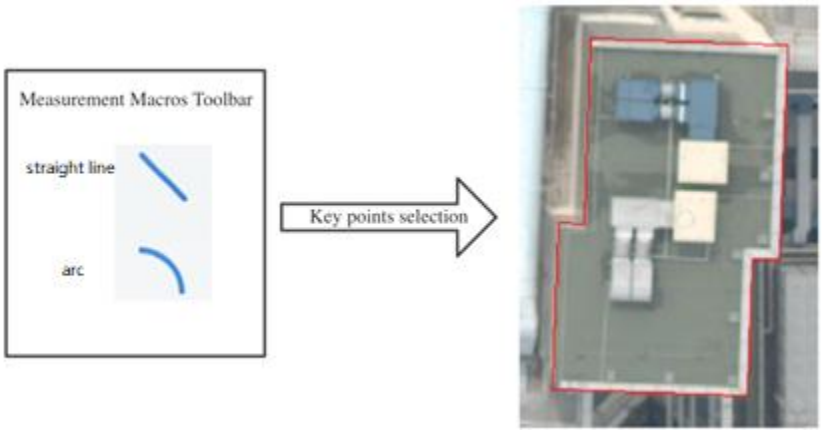
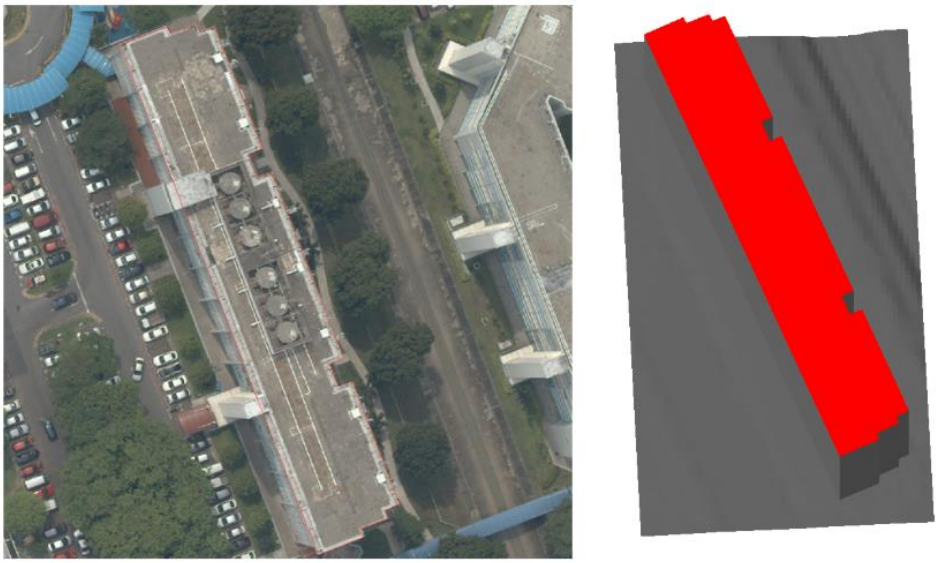



*Figure 5-33. Highlighted roof's boundary after a user has selected all the corners in an image*

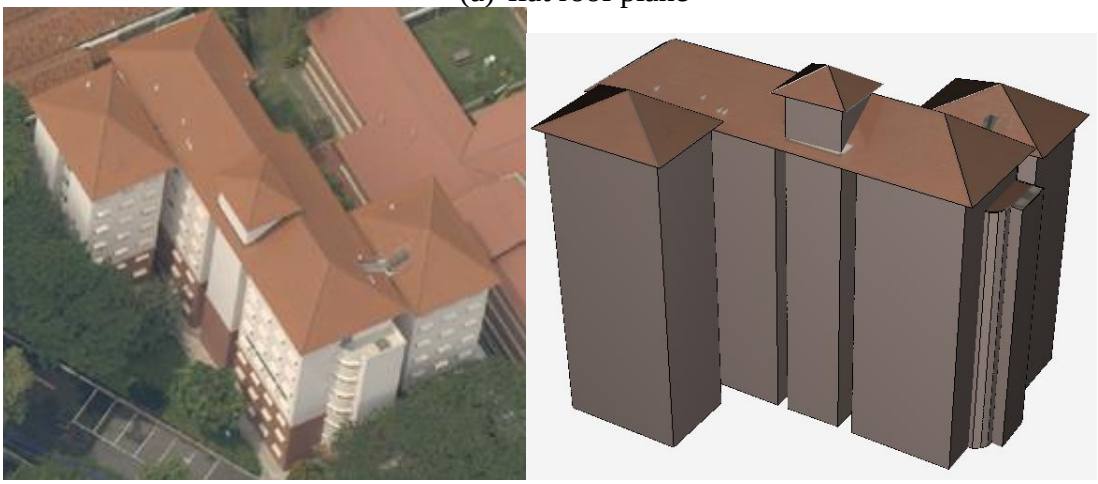
**b) Roof Planes Reconstruction**

Generally speaking, the roof planes determine the roof's main structure (Figure 5-34). At least three points are required to determine a unique plane in 3D space. Thus, users select three planar points in an image and repeat this step if there is more than one plane, all parameters of these 3D planes will be calculated and the intersections between these roof planes and the intersections between the roof planes and the roof boundary will be computed and modelled (Figure 5-35).

*Figure 5-34. Two typical roof styles in Singapore: a single flat roof plane (left image) and multiple roof planes (right image)*



(a) flat roof plane



(b) multiple roof planes

*Figure 5-35. Illustration of the roof reconstruction*

However, these manual works can be reduced when the points on the boundary are also the boundary points of the roof planes, which is often the case for the general buildings (Figure 5-35 (b)). Based on this investigation, a multi-planes fitting algorithm has been developed to assist operators to reconstruct the roof planes more efficiently, where operators only measure the inner keypoints in arbitrary order after the outmost boundary has been determined.
The proposed algorithm firstly computes the distances between each inner keypoint and all edges of the boundary, then groups all points into several coplanar consensus sets by the distance-based greedy strategy [5], finally solves the parameters of all planes simultaneously in a uniform least-squares framework [6]. The result of this algorithm applied to Figure 5-35 (b) is shown in Figure 5-36.

*Figure 5-36. Illustration of multi-planes fitting algorithm. Left image: the keypoints are marked with yellow circles while the outmost boundary is marked with yellow polygon; right image: each fitted plane is filled with different color.*

**c) Roof Parts Reconstruction**

There may remain some roof parts unmodeled. The measurement macros will be good functions to reconstruct the building part in some repeated pattern very efficiently. The measurement macros are combinations of several basic 2D/3D shapes, e.g. rectangles, circles, cubes, cones etc., to represent complicated building elements, such as balconies, windows etc. In such case we go through the following operator-executed steps:
Selection of area containing repetitive patterns > Selection of several key points to create a measurement macro > Entering of the vertical and horizontal element numbers; the measurement macro will be applied to the whole area. The basic idea is illustrated in Figure 5-37.

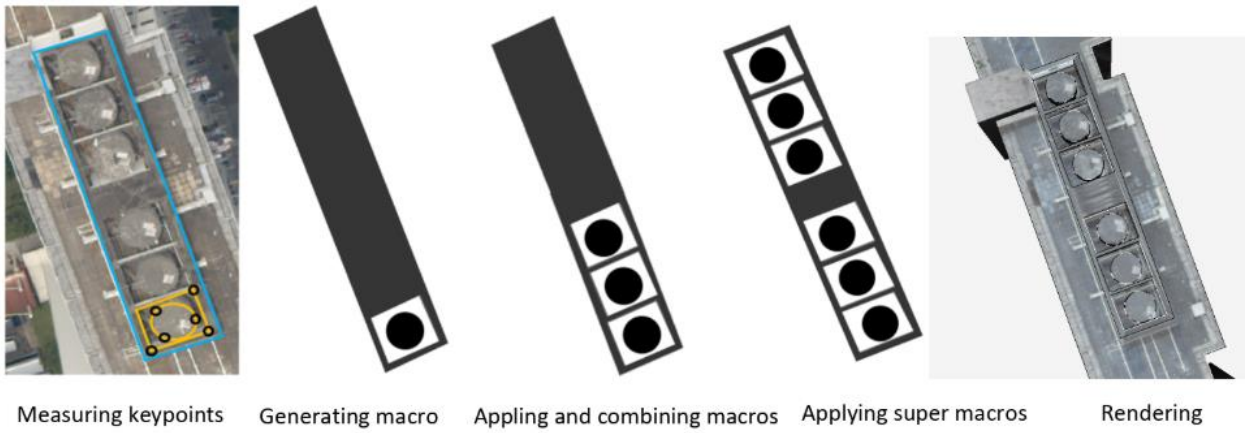
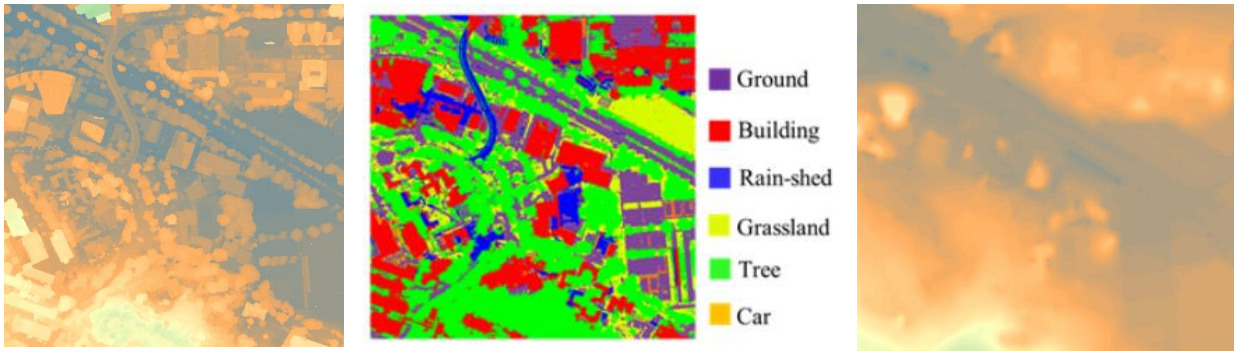



*Figure 5-37. Basic idea of measurement macros.*

### 5.1.3.4 Ground Surface Reconstruction

Generally speaking, the ground surface can be extracted from the DSM by filtering the natural and built features out. In our project, the DSM has two sources: one is from the 5-Head images processed by the Pix4Dmapper, the other is from the airborne LiDAR point cloud. To obtain an accurate and fine ground surface, the landcover classification algorithm developed by WP2 is firstly employed to the orthomosaic to get the initial regions of the natural and built features, the height difference detector is then applied in those initial regions to obtain their accurate locations in the DSM, the final accurate ground surface is achieved after the regions are replaced by interpolation (Figure 5-38).




(a) DSM (b) Classification (c) The ground surface

*Figure 5-38. Ground surface reconstruction. (a) the DSM generated from 5-Head images by the software Pix4Dmapper; (b) the landcover classification result from WP2; (c) the ground surface generated by filtering buildings, trees, and cars from DSM.*

### 5.1.3.5 Façade Reconstruction

Usually a building has at least four facade surfaces. These surfaces should be modeled one by one. As mentioned above, each line segment of the roof outermost boundary may correspond to a wall surface. Thus, in this procedure, the users choose a line segment of the roof boundary in our interactive software at first, then all related oblique images containing this line segment will be displayed for the following key points selection.

**a) Facade Surfaces Reconstruction**

Similar step to the above one “roof planes reconstruction”. Users should select three points from oblique images, but if introducing a restriction that façade surfaces are usually vertical, one less point is required, that is just two points not in a vertical line can determine the facade surface. After the facade plane is obtained, its intersections with the roof planes and the ground plane will be calculated to define the boundary of the facade surface.

**b) Façade Eements Reconstruction**

There are three types of façade elements which will be modeled, windows, doors and balconies. In this part, the measurement macros can greatly reduce the workload for operators since the windows and balconies are usually in some repeated patterns. For example, a facade full of windows is shown in Figure 5-39.

*Figure 5-39. A facade full of windows. The four corners of a window are selected by the operator.*

The operator first selects an area containing repetitive patterns, the black rectangle in Figure 5-39, then selects several keypoints to create a measurement macro (Figure 5-39 and Figure 5-40), finally enters the vertical and horizontal element numbers; the measurement macro will be applied to the whole area (Figure 5-40) in a copy-and-paste fashion

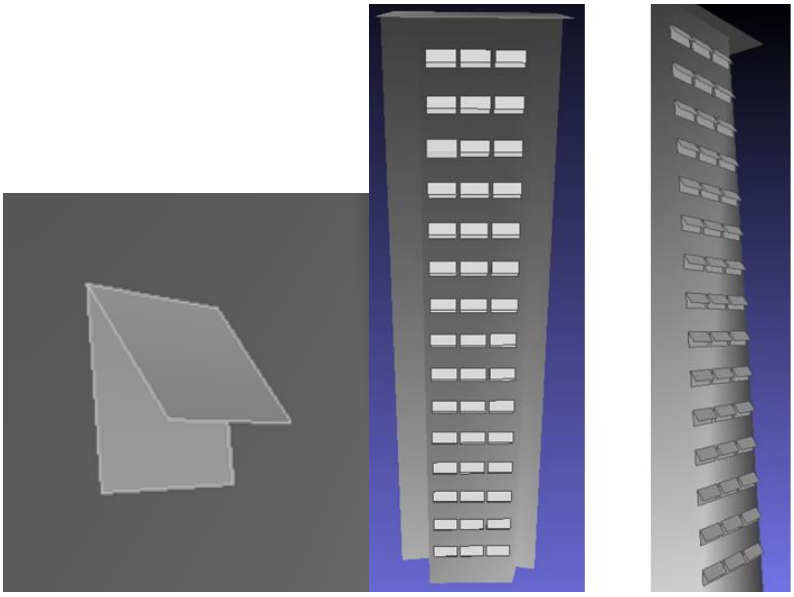


*Figure 5-40. Left: 3D model of the window after its four corners and 2 awning points have been measured. Right: Automatically generated windows after the operator enters the vertical and horizontal window numbers.*

### 5.1.4 3D Building Facade Segmentation

To ensure there is a maximal correspondence of the detected outcomes from 3D point cloud and the detection from aerial images, the targeting facade point cloud has to be located and segmented from raw point clouds. Since the huge size of the data, generally a raw point cloud data of a scene is separated in numbers of sets. In order to locate a specific facade in data sets, we first create a data management structure, indicating the path of the data and the scales of its area. This is simply done by computing the maximum and minimum points in the data.

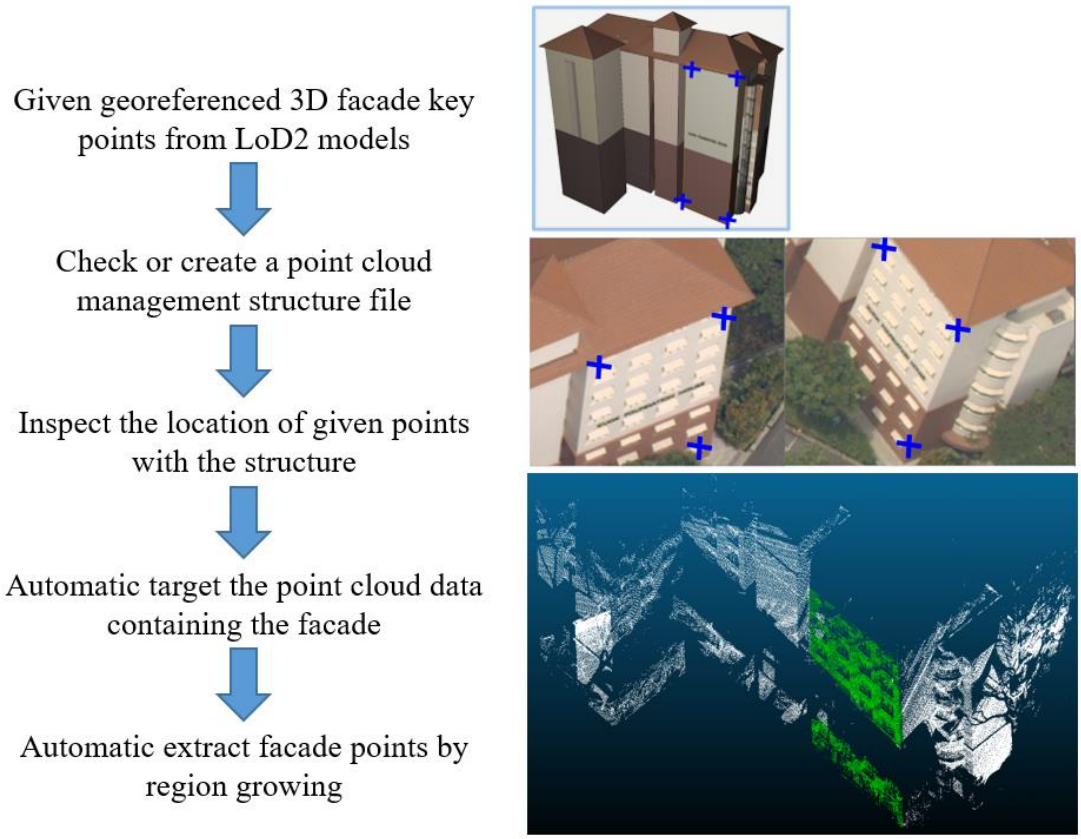



*Figure 5-41. The flowchart of automatic facade point cloud data targeting*

Given the georeferenced facade key points from LoD2 CityGML model that we generated using the above semi-automatic approach (see Figure 5-41 top right) or from direct measurement on images (see Figure 5-41 middle right) , we inspect the 3D coordinates of the given points with the data structure. If all the points representing the targeting facade are positioning in the area, the facade must locate in this point cloud data. Nevertheless, the detection algorithm relies on a single facade that has to be extracted from a point cloud data which contains it. The facade points can be extracted by region growing algorithm. We use the given key points as seeds and estimate a limited plane. For each point that is close to the plane and the growing seeds and also is within the limited area, it is labelled as a facade point. By this, the corresponding facade point cloud is ready (see Figure 5-41 bottom right) for the latter detection process.

## 5.1.5 A Sliding Window Method for Detecting Facade Features

### 5.1.5.1 Introduction

Architectural building models (LoD3) consist of detailed wall and roof structures including openings, such as doors and windows. Openings are usually identified through corner and edge detection, based on terrestrial LiDAR point clouds. However, singular boundary points are mostly detected by analysing their neighbourhoods within a small search area, which is highly sensitive to noise. We present a global-wide sliding window method on a projected facade to reduce the influence of noise. We formulate the gradient of point density for the sliding window to inspect the change of facade elements. With derived symmetry information from statistical analysis, border lines of the changes are extracted and intersected generating corner points of

openings. We demonstrate the performance of the proposed approach on the static and mobile terrestrial LiDAR data with inhomogeneous point density. The algorithm detects the corners of repetitive and neatly arranged openings and also recovers angular points within slightly missing data areas. In the future we will extend the algorithm to detect disordered openings and assist to 3D facade modelling, semantic labelling and procedural modelling.

Our approach of detecting openings corners is based on LiDAR point clouds of urban building facades. The facade point cloud collected by static or mobile LiDAR contains uncertain depths. This uncertainty will increase the difficulty for processing. For ease of the difficulty, we clip a rough primitive of the facade from the raw data. It is then projected to the local Y-Z coordinate plane. The approach consists of three main steps, which are elaborated on in the following.

#### 5.1.5.2 Sliding Window Search

Normally, facade elements are aligned following architectural rules. For instance, a slab is located between two rows or columns of openings when looking from the outside of a building. In other words, starting from an opening (e.g. a window), the area above belongs to a wall. Such rule is applicable in general standard buildings. To exploit this rule in the structured point cloud, we develop a sliding window method to inspect the facade horizontally from the bottom to the top and vertically from the left to the right

Taking the horizontal direction as an example, we first set up an initial window ($W_{hor}^{1}$as shown in Figure 5-42). The size of the sliding window depends on its depth, width and height. Since the window search is performed on the projected two-dimensional plane, in order to reduce the computational cost, we set the depth to 0 cm. The height is determined according to the pattern of the facade. As the prior knowledge of the architecture aspect, the slab between each floor is around 50 cm. Thus we assume that a striped window with the height of 40 cm (inner parts of the sliding window are equally 20 cm) is suitable to inspect the change of facade structures. In order to globally analyse the facade point cloud to reduce the noise effect, the width of the sliding window should be sufficient to cover the width of the facade data $Width_f = |Y_{max} - Y_{min}|$. To ensure all points can be counted, we slightly enlarge the width of the sliding window. The increase is half the height of the window. Therefore, the width of the horizontal sliding window is defined $Width_w = W_f + 1/2\ H_w$, where $H_w$donates the height of the sliding window.

After constructing the initial frame of the window, we start to define the sliding of the window. There are four key parameters need to be defined, including the starting and ending position, as well as the direction and distance of the slide. The position of the lower left corner of the window is marked as the starting position $(X_{hor},\ Y_{hor}, Z_{hor})_{start}$. In order to cover the entire area, the window based on this initial position needs to include the lowest and most marginal point $(X, Y, Z)_{min}$in the facade data. Therefore, we add a tolerance to this starting position, which is expressed as $(X_{hor},\ Y_{hor}, Z_{hor})_{start} = (X, Y, Z)_{min} - (Depth, Height, Height)_w/4$. Then, we give the direction of movement of the horizontal sliding window to the Z axis. It slides at half the height of the window each time $dist_{move} = H_w/2$. When it moves once, the initial position is updated according to the moving distance. The index of each point in the upper and lower parts of the window is stored for subsequent feature extraction and global information analysis. After recording all the information, the window will automatically slide along the direction and distance until it reaches the highest point $Z_{max}$ of the facade data.

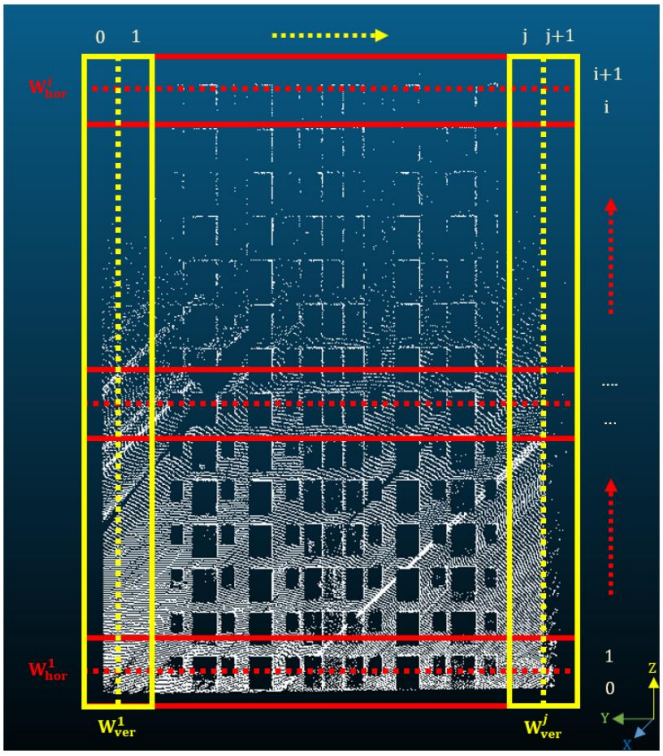



*Figure 5-42. Sliding Window Search. Red frames show the sliding window search in the horizontal direction. The yellow frame represents the vertical search window. Note that, for the sake of a clear visual representation, the size of the sliding window is enlarged in the figure.*

Similarly, for vertical search, the depth is still set to 0 cm. We set the width of the vertical sliding window to 40 cm. As half the width of the window, the sliding distance is 20 cm. The height of the vertical sliding window is defined as $H_w = H_f + \frac{W_f}{2}$, where $W_f$ is the weight of the facade. Assigning Y axis as the moving path, the vertical sliding window inspects the facade starting at the same location as the horizontal search until it traverses completely the facade.

### 5.1.5.3 Gradient Generation and Change Detection

Feature extraction from point clouds often rely on the relationship between a single point and its neighbourhood. This is sensitive to the noise surrounding the search point. Our proposed solution is to use a change of the number of points (see equation **(5-1)**) to establish a relationship to the overall area in the sliding window. Features, i.e. the change of facade elements, can be extracted by these gradient values and the corresponding window positions.

$$\mathbf{G} = \frac{\mathbf{N}_{i+1} - \mathbf{N}_i}{\mathbf{N}_{i+1} + \mathbf{N}_i} \rightarrow \begin{cases} 1 & \mathbf{N}_i \ll \mathbf{N}_{i+1} \\ 0 & \mathbf{N}_i \approx \mathbf{N}_{i+1} \\ -1 & \mathbf{N}_i \gg \mathbf{N}_{i+1} \end{cases} \quad \textbf{(5-1)}$$

Following the mentioned architectural rule, it is assumed that the number of points within the sliding window will be reasonably large when it moves to the slab or other place that belongs to the wall. On the other hand, when the area inside the sliding window belongs to openings where basically no representing points, the amount of points will be relatively small. We compare the points within the upper and lower parts (left and right in the vertical direction) of the sliding

window. If the sliding window arrives at the boundary between slab and openings, the number of points in the two parts of the window will change greatly. Therefore, based on the number of points acquired in the previous search, we extract features by calculating the gradient of the number of points in the sliding window.

However, most facade point clouds have inhomogeneous density. For example, due to the scanning distance and occlusions, the density of points in the upper part of the facade is less than that in the lower part. This challenge will increase the uncertainty of the range of gradient changes. To alleviate this problem, we assume that the property of points located in the current sliding window is similar, such as they have approximate point distances. Therefore, we can effectively compress the gradient values to the 0 to 1 interval by dividing the difference between the points on the upper and lower parts of the sliding window by the number of points, which is expressed as

Where $N_i$ means the number of points in the upper (left) part of the sliding window; $N_{i+1}$ is the number of points in the lower (right) part. These two numbers are positive. When G approaches to 1 that means the number of points in the latter area is much larger than the former one. Conversely, a negative value means that there are more points in the first half. When G approaches 0, it means that the points of the two parts of the sliding window are similar.

The sign of the gradient G indicates the variation trend of the facade elements. Since the number of points must be non-negative, the sign of the gradient depends on the difference between the first and second parts of the sliding window. When the search window slides from wall to openings, the gradient of the number of points will have a change from positive to negative. However, this process is subject to a certain degree of noise, which is caused by the points from non-interest objects. For analysing the change position and noise influence, we plot the gradient values as shown in Figure 5-43.

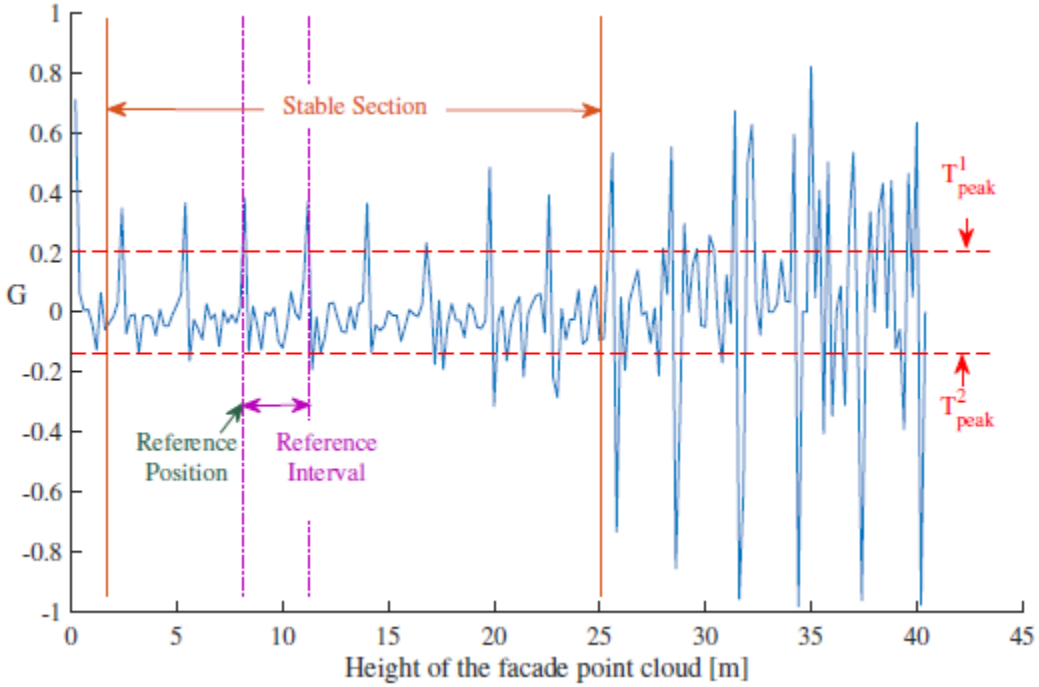



*Figure 5-43. Statistical analysis of gradients G in sliding windows*

Following the assumption, the peaks of the gradient in the statistic figure generally represent the change between openings and wall. Such peak values can be extracted by two filters. $T^1_{peak}$

restricts the positive values (e.g. a window to a slab) that values below this threshold will be filtered out. In contrast, the gradients staying over the constraint will be treated as peak values. Similarly, we keep values which are less than the $T_{peak}^{2}$ constraint (e.g. a wall to an opening). Each change should perform a sharp rising or down trend following a regular pattern.
However, due to the low point density and other effects, the performance of gradients becomes unstable in the higher part of the facade. To reduce the impact, we predefine a stable section to extract symmetric information to constrain the overall region. First, we calculate the interval between all extracted peaks and sort them in descending order. Then, the mean $\underline{I}_{peak}$ and median $median\ (\boldsymbol{I}_{peak})$ of the peaks in the given stable section are calculated. By comparing these two numbers, the smaller one is taken as the symmetric information, representing the reference interval in Figure 5-43. The definition is expressed as

$$I_{ref} = min\ (\underline{I}_{peak}, median\ (I_{peak})) \quad \textbf{(5-2)}$$

#### 5.1.5.4 Corners of Facade Openings

The final step in the approach is to intersect the corners of openings by extracting the horizontal and vertical intersection of the wall and the openings. From the position of sliding window given in the previous step, we can locate the boundary between the wall and openings. Since each gradient value is obtained from the points corresponding to the upper and lower parts of the sliding window, in order to obtain robust borderlines, we use the intersection line of the two parts to approximate the boundary between different facade elements.
Empirically, the bottom and the top of the facade are not in the stable section, due to occlusions and noise effects. It will cause misleading boundaries if we extract them following the search order of the sliding window. Thereby, we assign a reference peak position. The extraction of horizontal (vertical) borderlines is executed from the referenced position to the bottom (left) and to the top (right) separately under the constraint of the derived reference interval $I_{ref}$. Finally, corners of openings can be generated by intersecting the horizontal and vertical boundary lines.

### 5.1.5.5 Summary

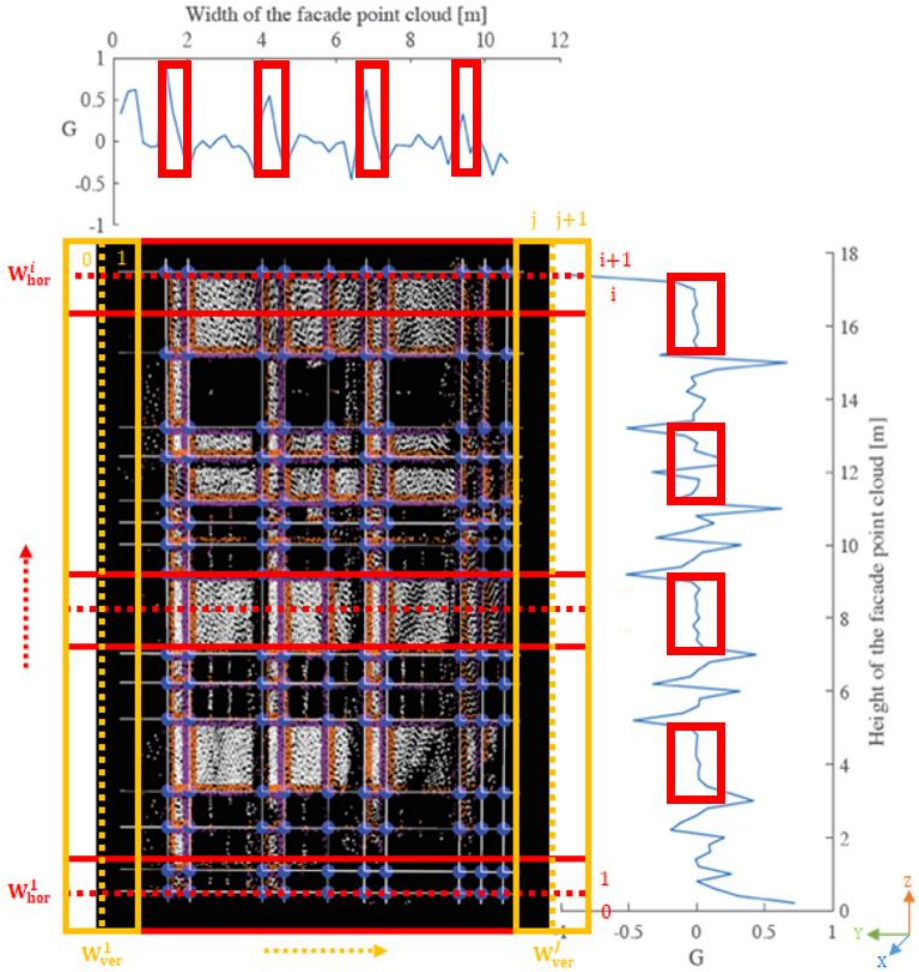



*Figure 5-44. Facade feature extraction using the sliding window search. Red frames on the feature space placed at the right and top show the facade features corresponding wall surface. Other features such as spacing between openings, size of openings, size of the facade and possible layer of buildings can be inferred from the detected features.*

We propose an approach to detect corner points of repetitive openings from terrestrial LiDAR point clouds. The approach searches and analyses the facade with a globally wide sliding window that mitigates the noise effect in horizontal and vertical directions based on octree structure. Through computed gradients in each sliding window, we apply statistical analysis and derive the symmetry information to inspect changes of facade elements. This information can be parameterized to procedural modelling in our latter process. The approach detects corners of openings and recovers corner points in a partly missing data area. However, mostly occluded areas and not aligned openings are the limitations.

## 5.1.6 Opening Detection

Openings such as windows and doors are essential components to architectural wall surfaces and yet still being a challenge for robustly reconstruction from unstructured 3D point clouds. Current research principally focuses on meliorating the robustness of detection and pays little attention to

the connectedness and global representation of openings which potentially enrich building information, such as three-dimension interoperable geometry and structural information. In this paper, we propose a novel and resilient approach for inferring the topology structure and layout of wall surfaces through robust opening detection and flexible rule inference across multi-layer. Our method first detects openings on data-adaptive reference coordinate system and optimizes the geometric aspect of the reconstruction. Employing the reconstructed marks, it infers the spatial relationship between objects based on flexible alignment rules and topological relations of opening associations. As our method is oblivious of the type of architectural layout, it can be applied to interior wall surfaces and exterior building facade. We demonstrate the flexibility of our approach in both outdoor and indoor scenes with a variety of opening layouts. The results indicate the proposed engine is potentially to be a general use in opening detection and association.

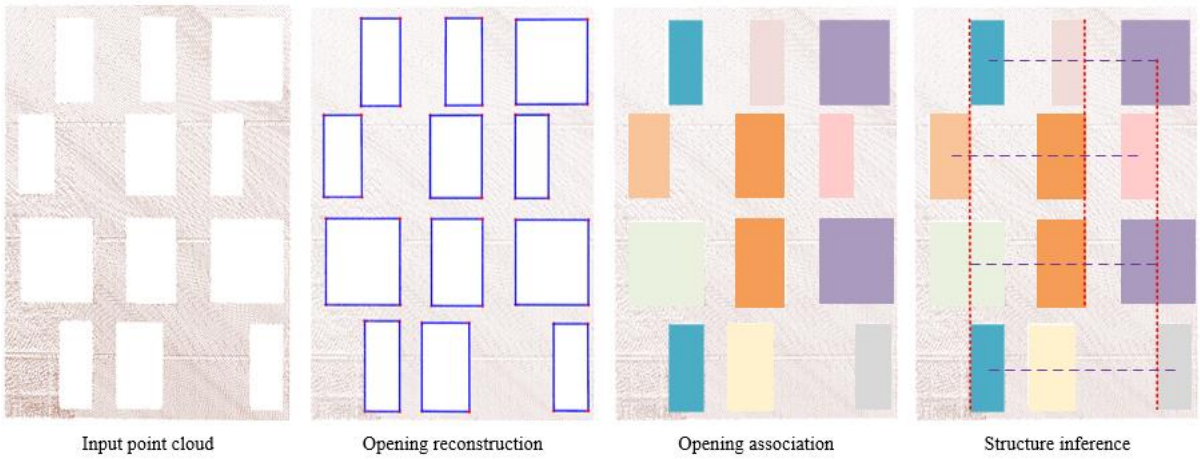



*Figure 5-45. Reconstructing geometry and topology of openings on a wall surface point cloud. From left to right: Input 3D point cloud with a designed layout of architectural openings, detected and reconstructed the geometry of openings, inferred the association based on detected flexible rules*

To ensure a robust detection of openings (e.g. windows and doors), we first detect the targeting wall surface from a raw point cloud data. This pre-processing step can be done by the approach which is mentioned in section 5.1.4 (Building Facade Segmentation). The input wall point cloud has an uncertainty of facing direction. We evaluate all the wall data based on the estimation of corresponding planar surface. A reference coordinate system is defined and set up for the whole process of the inference machine. Then we demonstrate the process of initial opening detection and reconstruction as shown in Figure 5-46.

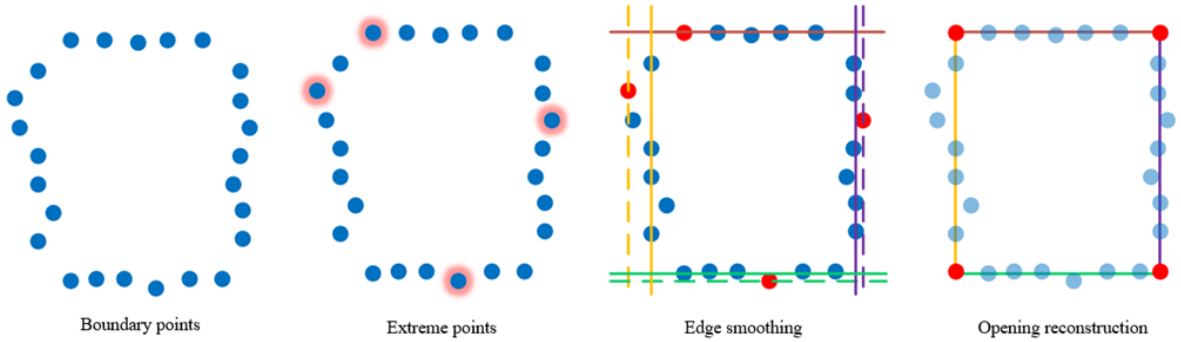



*Figure 5-46. Overview of the pipeline for opening reconstruction. From the left to the right: we detect boundary points of each single opening. Then we extract extreme points and optimized them by smoothing energy function. A rectangular shape of opening is then reconstructed which is represented by the four ordered corner points in red.*

### 5.1.6.1 Definitions

We define the plane **N** of input wall surface data using a standard plane parameterization as

$$\mathrm{N} = \begin{bmatrix} \hat{n} \\ d \end{bmatrix} \in \mathbb{R}^4 \qquad \textbf{(5-3)}$$

$n$ is the unit normal of the plane and $d$ is the scalar which represents the closest approach of the plane with the origin. We use RANSAC to estimate the unit normal n^_f of the facade plane $N\_f$. In order to determine the horizon and vertical axes, $n\hat{}\_h$ and $n\hat{}\_v$ respectively to the reference system $\mathrm{S_r} = (\hat{n}_h, \hat{n}_v, \hat{n}_f)$, we insert an x-y plane defined in Cartesian coordinate system $\hat{n}_z = (0,0,1)$ to the planar wall surface that the horizontal direction $n\hat{}\_h$ is defined as $\hat{n}_h = \hat{n}_f \times \hat{n}_z$. Similarly the vertical axis $\hat{n}_v$can is generated by a cross production of facade normal and the horizon $\hat{n}_v = \hat{n}_f \times \hat{n}_h$.

### 5.1.6.2 Boundary Extreme Point Detection

Given the above definition, we transform points to the plane of wall surface by a projection function $\pi_f(\cdot): \mathbb{R}^3 \rightarrow \mathbb{R}^2$. On the defined planar surface, we exploit Alpha-shape [60] to extract candidate boundary points of each opening. Let a 2D polygon Ω be given by n vertices $\Omega = \{v_0, v_1, \dots, v_{n-1}, v_n = v_0\}$ ordering anti-clockwise around the polygon.

The idea of extracting the extreme points of a polygon is to search the corresponding maximum and minimum points on arbitrary projection line along with two directions on the plane. The projection function is defined as $\pi_d(.): \mathbb{R}^2 \rightarrow l$. Since the size of polygon Ω is indeterminate to each opening due to the size of the object and also the density of the raw point cloud data, we use binary search to locate the maximum and minimum points on each projection line. The extreme points of the polygon detected using alpha-shape can be formulated as:

$$\mathrm{M} = \left\{ \underset{d\in\{\vec{n}_h,\vec{n}_v\},\, v\in\Omega}{\arg\max}\ \pi_d(v),\ \underset{d\in\{\vec{n}_h,\vec{n}_v\},\, v\in\Omega}{\arg\min}\ \pi_d(v) \right\} \quad \textbf{(5-4)}$$

### 5.1.6.3 Edge Smoothing (Sliding Window)

The extractions can simply represent a bounding box to the object. Nevertheless, due to slightly occlusions or measurement error during data acquisition, initial extreme points $\mathbb{P}^m = \{p_0^m, p_1^m, p_2^m, p_3^m\} \in M$ can be hardly used for a proper approximation of the configuration of opening. In order to ease this issue, we introduce an objective function for edge smoothing which is expressed as:

$$E_s(\mathbb{P}^m) = E_p(\mathbb{P}^m) + E_l(\mathbb{P}^m) \quad \textbf{(5-5)}$$

with

$$E_p(\mathbb{P}^m) = \sum e^{-\xi(\mathbb{P}^m)}$$

and

$$E_l(\mathbb{P}^m) = \sum_i \sum_j d(P_j, l_i(\mathbb{P}^m))$$

The objective of Equation (5-5)**.** is to optimize the extreme point set $P^m \rightarrow P^{m'}$. Since the initial extreme points $p_i{}^m \in P^m$ are supposed to be the maximum bounding points of the corresponding object, the optimum shape should be equal or within the current configuration. Thus, we shrink the parametric frame towards the centroid of an object subjected to two energy terms. The first term smoothes the edge $e(p_i{}^m, n)$ by measuring the neighborhood point density $\xi(.) = |P_e|$ subjected to the constraint of the neighboring distance $dist(p, e) < \varepsilon$. $\varepsilon$ is an empirical value which is set to half of the alpha value $\alpha/2$. The second term is similar to the polygon smoothing in O-Snap [61] that prevents the bounding edge from over deviation from the initial position. It avoids over shrinking from the origin configuration of objects. As the number of openings and models is generally small, we minimize the optimization function using *Best-first search* [62].

### 5.1.6.4 Opening Reconstruction

Once we finalize the optimum extreme points $P^{m'} = \left\{P_0{}^{m'},\ P_1{}^{m'}, P_2{}^{m'}, P_3{}^{m'}\right\} \in \Omega$, the configuration of the object is basically approximated, such as the location, orientation and initial topology. However, for the seek if the correlation between detected object in the following stage, we need the object to be represented in rectangle shape. To frame such mark of shape, We intersect lines [63] defined by the reference direction (Section 4.1 and the optimized extreme points $P^{m'}$. Corners of the rectangular mark are detected at the intersection points of two lines in perpendicular directions. Note that, the reconstruction is anti-clockwise starting from the left lower corner (e.g. the intersection of left and bottom line) of the rectangular shape.

### 5.1.7 Opening Association

From the above process, we detect and reconstruct discrete openings marked in rectangular shape from wall point clouds. In this section, we show a further process to infer the spatial topological relationship based on the marks. The idea is to identify implicit rules between reconstructed openings. We observe that in general a designed layout of openings on a wall surface can be decomposed and reconstructed by a set of flexible rules. These rules also reveal the relationship between openings. For instance, windows perform in a most common grid pattern on a building facade. The regular pattern is comprised of fundamental elements of horizontal and vertical lines. Foundations such as alignment line rules we proposed in the following can be detected and used to flexibly associate correlative wall elements. With the rules and associations, we then infer the correction to the opening and the connection to the groups. It reveals that all the relationships including the association of openings and the connection of the associations are hierarchically inferred based on the resilient rules. We therefore introduce the inference machine in three steps.

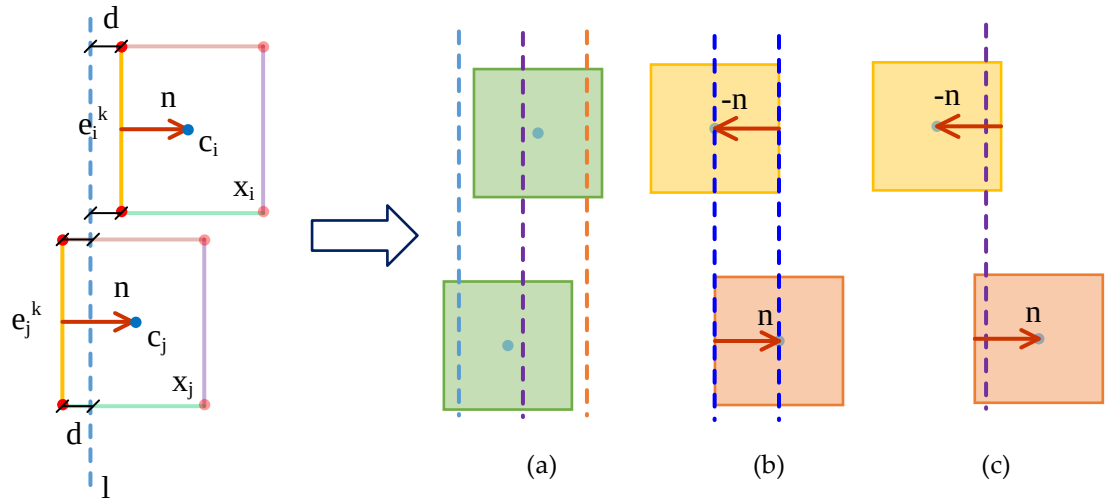



*Figure 5-47. Opening association inference. (a) Example shapes in green are associated because of the alignment to the left, center and right within a predefined tolerance. (b) Association failed since the center-left and right-center alignment differ in the vector pointing to the centroid. Similarly, in (c), right-left aligned with opposite normal vectors fails the association (marked in yellow and pink respectively).*

#### 5.1.7.1 Flexible Rules Generation

In order to increase the degree of freedom to rule detection, we use nodes including the vertices $\Psi = \{v_k \mid k = 0,1,2,3\}$ and the corresponding centre point $c$ on the first layer. Let $X = \{x_0, x_1, x_2, \dots, x_n\}$ be the configuration set of detected openings. For each opening $x_i = \{\Psi^i, c^i\}, \forall x_i \in X, i \in \{0,1,\dots,n\}$, where $\Psi^i = \{v^i{}_0, v^i{}_1, v^i{}_2, v^i{}_3\}$ donates the vertices of $x_i$. The alignment of objects can be transformed into the alignment of vertices. To distinguish points align to a rule, i.e. a line, we first index potential alignment clusters of the points in $X$. Similar to the previous step in searching extreme points, we project all the points to horizontal and vertical lines on the plane defined in the reference coordinate system $S_r$ using the projection function $\pi_d(.): \mathbb{R}^2 \to l$. For clustering the projections, we use an Euclidean clustering approach [58] that let $\pi_d(X)_i = \{p_i \in P\}$ be a distinct point cluster from $\pi_d(X)_j = \{p_j \in P\}$ if $min\ ||p_i -$

$p_j||_2 \geq d_{th}$ . Then initial alignment rules (here we define a line as a rule) are estimated from clustered points by RANSAC.

### 5.1.7.2 Opening Association

Let's assume $L = \{l_o, l_1, l_2, \ldots, l_m\}$ be a set of rules we detected on the previous step. Lines $L$ indicates all possible associations of openings $G = \{x \mid dist(v^i{}_k, l_m) \leq d_{th}, x \in X, k \in \{0,1,2,3\}\}$ while containing a plethora of information against our goal of the maintenance of the relationship between objects and their associations. Therefore, we infer initial opening groups sharing connection rules on the 2nd-layer by a two-term condition function as the following:

$$E_g(G) = \lambda_1 E_c(G) + \lambda_2 E_f(G) \quad \textbf{(5-6)}$$

The first term $E_c(G)$ introduces the consistency of neighboring openings which attach to common lines. It also considers the alignment of the openings by calculating the dot products of two neighboring normal vectors $n^i(e^i{}_k, c^i)$ (w.r.t $n^j$) pointing from the edge $e^i{}_k$ (w.r.t $e^j{}_k$) to the corresponding centre $c^i$(w.r.t $c^j$) , where $e^i{}_k, c^i \in x_i \in l_m$; $e^j{}_k, c^j \in x_j \in l_m$ . The consistency term is expressed as:

$$E_c(G) = \sum n^i(e^i{}_k, c^i) \cdot n^j(e^j{}_k, c^j)$$

*Fidelity term*

$$E_f(G) = \sum \omega \cdot d(e, l)$$

with

$$\omega = (1 - \mu)\frac{|s|}{|l|}$$

The second term describes the fidelity of object groups relied on the line rule. We measure the average distance d (see Figure 5-47) between the line rule and endpoints of edges which indicates the tolerance of adjustment.  donates the probability of the distance against the rule. We use the ratio of the quantity of representative elements contained in a rule to indicate the attraction of the line. The higher number of stable elements the rule contains, the higher fidelity the line rule has. The edge with a certain distance to the trusted rule has higher probability to be adjusted close to the line. Moreover, in order to ease the impact from imprecise rules, we add a confidence value =0.05.

We solve this energy function again through Best-first search. We aim at the search of rectangle marks with similar attribution through the alignment to their edges. Therefore, the first energy term describing the correspondence of alignments is vital and it is the foundation of the inference machine. As demonstrated in Figure 5-47, if the dot product between the two normal vectors $n^i$ and $n^j$ is positive and approaches to 1, means two edges are aligned, then $E_c(G)$ is set to 1. Interrelated objects are classified into a temporary group. Otherwise, it turns to -1 and split entities into two different groups. Distinction of normal direction will be conducted simultaneously in the split groups for the rest of candidates. In order to increase the degree of freedom to the rules, we introduce center alignment to the inference machine. The dot product is

close to 0 when center-left, center-right, as well as center-top and center-bottom alignment happen. In such cases, it will create a new provisional group for the center alignment. In our experiment, we set $\lambda_1 = 1.0$and $\lambda_2 = 0.5$.

#### 5.1.7.3 Association Refinement and Connection

From the last step, we obtain associations of openings by inferring multiple rules. Comparing to groups attached to each line rule, a few updates can be applied to those temporal groups by exploring the repetition of group elements. The function in this step can be deemed as an overlay of the previous two with entire flexible rules and the inferred associations respectively. On this, we aim to refine the outputs and extract the chaining rules for all these associations toward a reconstruction of facade structure. It is able to observe elements within initial groups in the line rule are in a misallocation. Synchronously, the line which link the most groups is selected as the relevant connection between the associations.

#### 5.1.7.4 Results

We apply the resilient inference machine to 3D point clouds of outdoor and indoor scenes. Figure 5-48 shows an outer surface of a residential building captured by mobile mapping system. It is obvious that the completeness of data is not perfect as most of the realistic data that might be occluded by trees or surrounding objects to the sensor. From our observation, it still allows us to reasonably approximate the configure of the facade openings and reconstruct its layout. With the segmentation of the target facade plane**,** we found the basic structure of the facade, i.e. boundary points of openings, remains in a sufficient quality that allows the extraction by the alpha-shape algorithm. Since the point density is sparse in this data, we need to release the neighboring point distance to the algorithm for obtaining proper extreme points. In such case, we set alpha value to be 0.28. Furthermore, each red line shown in the figure threads an association of openings which is inferred by the proposed flexible rules. For instance, the first row of the openings belongs to a group as well as the third column (from the left to the right) entities are grouped. This figure illustrates that our flexible rules are able to reconstruct the layout and structure of facade openings (here it overall presents a grid pattern), simultaneously, retain the spatial relationship of the objects.

The facade openings in static terrestrial LiDAR point clouds in Figure 5-49 are successfully detected and associated into several linked groups. Note that the machine infers associations and accordingly updates the geometry of openings (see the upper left of Figure 5-49.), whereas the occlusion in the raw data influences to the final alignment (as shown in the lower right of Figure 5-49*).*

Our pipeline is also applicable to indoor environment. We demonstrate it on ISPRS datasets [64]. Note that, for the sake of a clear visual representation, the raw point cloud in these figures are subsampled. Figure 5-50 demonstrates the flexibility and adaptability of our proposed resilient inference machine. The approach is applied to every vertical faces within the original coordinate system. It ensures the robustness to the detection on arbitrary planar surfaces. Half opened windows or doors, such as the door in the middle of the data, can be detected according to the opening angle.

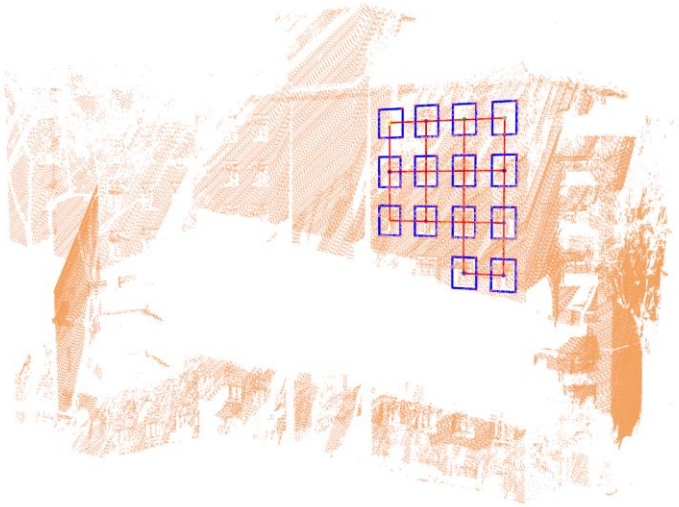


*Figure 5-48. Facade opening and layout reconstruction on mobile LiDAR point cloud data. Blue frames represent detected openings with the association representing in red lines.*

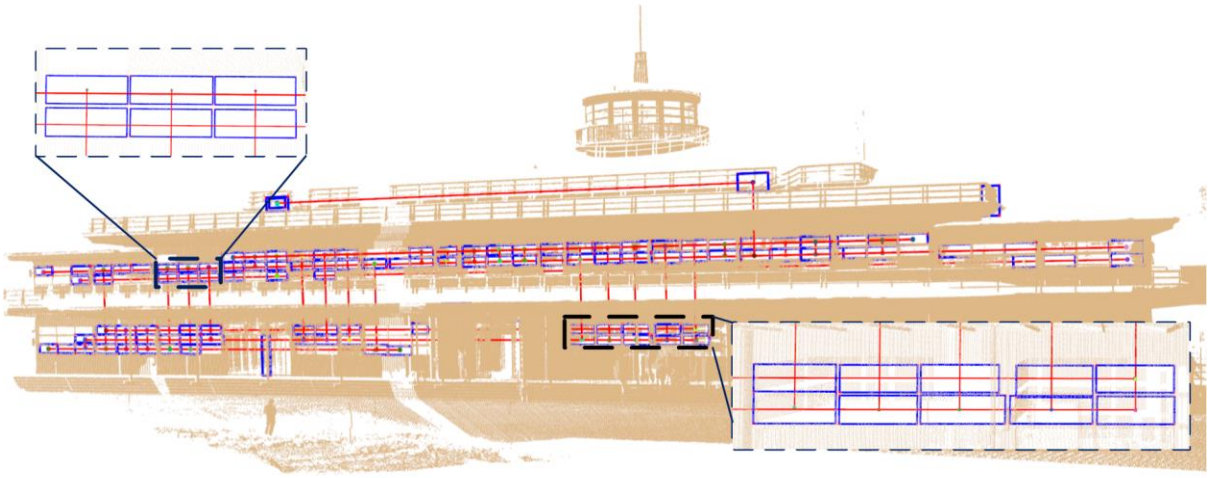


*Figure 5-49. Facade opening detection and association on a static terrestrial LiDAR point cloud. Upper left the optimization of alignment and geometry of openings. Lower right of the figure illustrates the influence to the optimization by* occlusions.

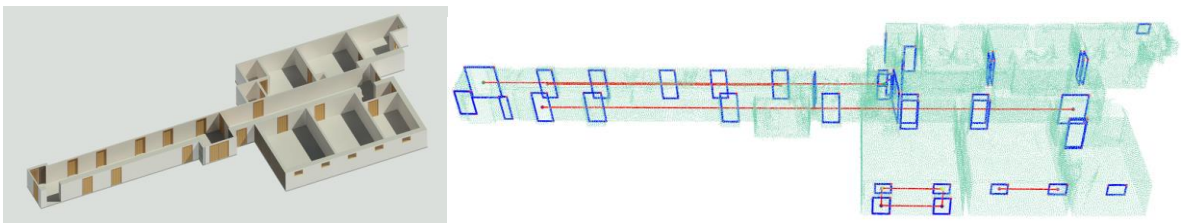


*Figure 5-50. Opening detection and association in the point cloud of TUB1. Left: benchmark model of TUB1. Right: Reconstructions (blue frame) and topology (red line) of openings (doors and windows) on wall surfaces. For a sake of clear presentation, the TUB1 point cloud is subsampled by minimum space of 0.1m.*

## 5.2 WP2: Land-cover Classification, Semantic Enrichment and Texture Mapping

### 5.2.1 Landcover Classification

#### 5.2.1.1 Introduction

Land-cover classification is an intensive research topic in the remote sensing community and has been widely investigated for decades [7, 8]. The top view images (e.g., orthographic satellite images and orthophotos of photogrammetric products) as conventional approaches for land-cover classification, are standard and well-practised [9]. However, high intra-class variability and inter-class similarity are making the classification difficult, especially for man-made objects in urban areas. These man-made objects can be quite confusing at the top view, such as buildings with green/sports roofs compared to grasslands and playgrounds. To better classify these confusing objects, the elevation information from a georeferenced digital surface model (DSM) or 3D point cloud has been analysed and involved in current studies [10]. Yet, the height information can relieve the confusions at flat grounds, it is fragile at the terraced areas where the above-ground heights are hard to be acquired. In this study, to complement the insufficiency of these top view based features, we seek to the side views captured by oblique imagery.

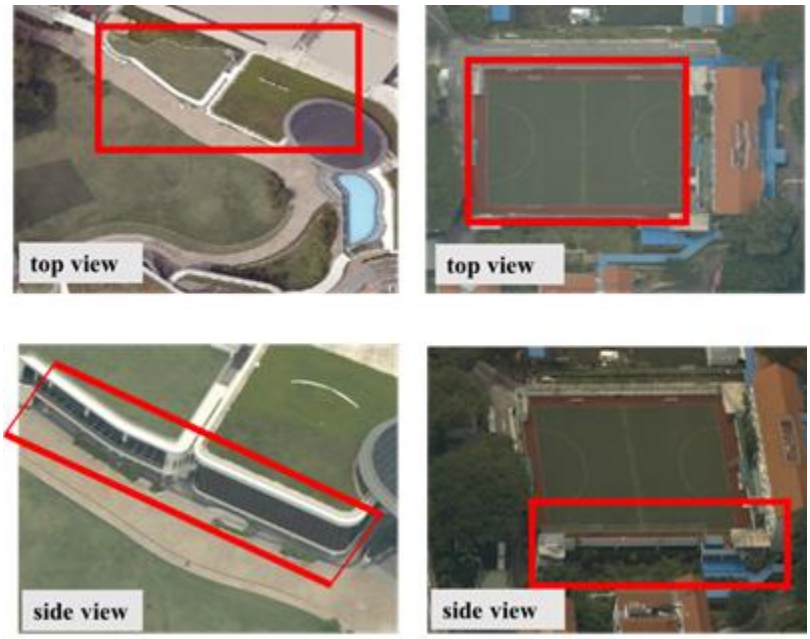



*Figure 5-51. The side view information from the oblique image can help the land-cover classification.*

As illustrated in Figure 5-51, side view information is helpful to distinguish the above-ground objects with similar top views. However, to find and attach the vertical side views to their hosts in the orthographic top view images, could be a knotty problem. There is no direct connection between a region in the top view image and its possible side textures in the oblique images, even we schematically linked them in Figure 5-51.

Recently, with the development of photogrammetric techniques, oblique images are widely used for 3D reconstructions with mapping products, such as orthophoto and DSM that have been

extensively analysed for the land-cover classification [10]. By observing the geometric constraints between the oblique images and the orthophoto and DSM, finally, we find a way to incorporate the side view information for the land-cover classification. Firstly, from the DSM, the above-ground objects can be segmented out as the regions that could have side view information. Then, for each above-ground object, a virtual polygon boundary would be calculated to map the side view textures in the oblique images via a perspective transformation. Finally, the side view features can be extracted from these textures and incorporated in each above-ground object segment for the classification with their top view features.
Following this idea, in this project, we aim to leverage the extra side view information to improve the land-cover classification with the oblique imagery derived orthophoto and DSM.

#### 5.2.1.2 Side View Information Extraction

To incorporate side view information in land-cover classification, firstly, we segment out the above-ground objects which could have side views at the individual level. Then, built on the segmentation boundaries, their side views are mapped and selected from oblique images via a perspective transformation. Finally, colour and texture features from the side view textures are extracted for the classification of each above-ground segment.

**a) Above-ground Object Segmentation**

Above-ground object segmentation is a complicated problem which has been studied for decades [11, 12], but still does not have a general solution. To simplify this problem, we assume all above-ground objects have flat roofs, for example, if a building has two conjoint parts with different heights, then the two parts are treated as two above-ground objects. This assumption can help us to efficiently segment the above-ground objects at the individual level by a simple height clustering algorithm, in which the connected pixels that share similar height are grouped as one above-ground object. To implement this clustering, firstly, we use the DSM to calculate a gradient map which can approximately represent the pixel height of above the surrounding pixels. Then, from the highest to the lowest, the connected pixels with height differences in 1 meter are sequentially grouped as individual segments, as shown in Figure 5-52. Finally, the segments which have 2.5-meters average above-surrounding heights are classified as above-ground objects.
In some cases, this clustering may contain errors, such as uncompleted segments and misclassifications, which are mainly caused by the disunity of the height (e.g., the towers on the roof and the gullies on the ground). To fix these errors, region merging and above-surrounding height check are utilized as the post-processing.

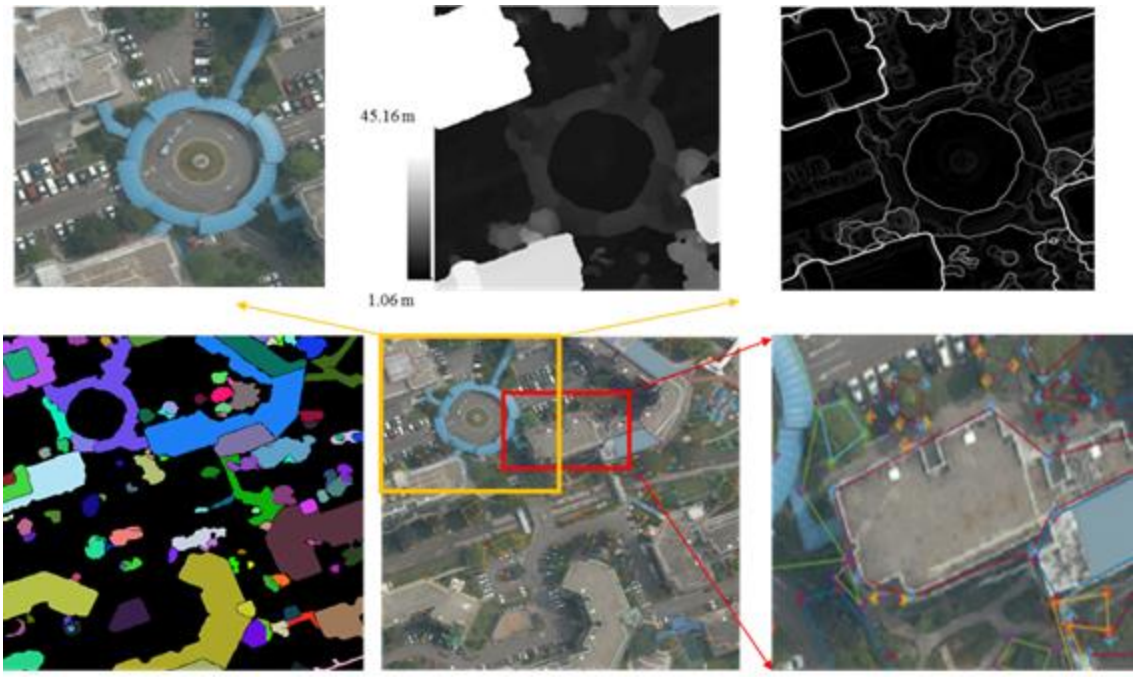



*Figure 5-52. The segmentation and polygon boundary generation of above-ground objects. The upper images are the orthophoto, the DSM and the gradient map. Lower images are the segmented above-ground objects showing in different colours and their polygon boundaries.*

**b) Side View Texture Cropping and Selection**

Similar to the 3D building façade texture mapping work [13], the vertical faces of above-ground objects can be mapped and cropped from oblique images. However, unlike buildings which have a clear definition of the façade plane, many above-ground objects, for example, trees do not possess a specified vertical face. To solve this problem, we convert the boundaries of above-ground segments into polygons by the Douglas-Peucker algorithm [14], thus to create pseudo vertical faces by dropping polygon lines from the top of the object to the ground (in the experiment, only the longest three lines are used to extract the side view images). As illustrated in Figure 5-53, the vertical face is defined as a rectangle with four space points (P1, P2, P3, P4). The upper points (P1, P2) are the two ending points of a polygon line with the object height, while the lower points (P3, P4) are the same points but with ground height. The 3D coordinates $(X, Y, Z)$ of the four points in the object space can be acquired from the georeferenced orthophoto and DSM, thus their corresponding oblique image coordinates can be calculated via a perspective transformation:

$$s(u, v, 1)' = P_{3\times4}(X, Y, Z, 1)', \tag{5-7}$$

where $(u, v, 1)$ is the 2D homogeneous coordinates in the oblique image with  as a scale factor, and $P_{3\times4}$ is a perspective transform matrix which contains the intrinsic and extrinsic camera parameters that are calibrated before the photogrammetric 3D processing. The reader can find more details about the photogrammetry in [2]. As illustrated in Figure 5-53, after this perspective transform, the four points can define a region of the side view in the multi-view oblique images. To get better side views for the later feature extractions, we rectify the textures to the front view with a homography transform.


*Figure 5-53. The side view texture extraction from multi-view oblique images. From left to right, the images are the polygon boundaries, the 3D vertical face and its projections at the oblique images, and the side view textures with a rectification. More details are explained in the text.*

As illustrated in Figure 5-53, usually, there are more than one oblique images can capture a side-view of an object. To effectively choose the best one, we consider three factors: 1) V(f), the quality of the angle between the normal of the face plane and the camera imaging plane, 2) N(f), the quality of the angle between the face normal and the line through camera and face centres, 3) O(f), the proportion of the observable part. Based on these factors, the best side-view is selected by a texture quality measurement:

$$Q(f) = m_1 * V(f) + m_2 * N(f) + m_3 * O(f), \tag{5-8}$$

where Q(f) measures the quality of side view $f$ , while $m_1, m_2, m_3$ are the weights of different quality factors. In the experiment $m_1, m_2, m_3$ are set as 0.25, 0.25, 0.5, respectively, as we found the observability is more important. Besides these factors, the size of façade texture (texture resolution) is important in texture mapping. However, we did not consider it in this study, because a high resolution texture is not crucial for the side-view feature extraction. While the first two factors can be easily calculated, the observability is complicated to be measured due to the occlusion which commonly exists in urban areas. Inspired by a Z-buffer based occlusion detection [15], we examine the observability with a distance measurement as illustrated in Figure 5-54.

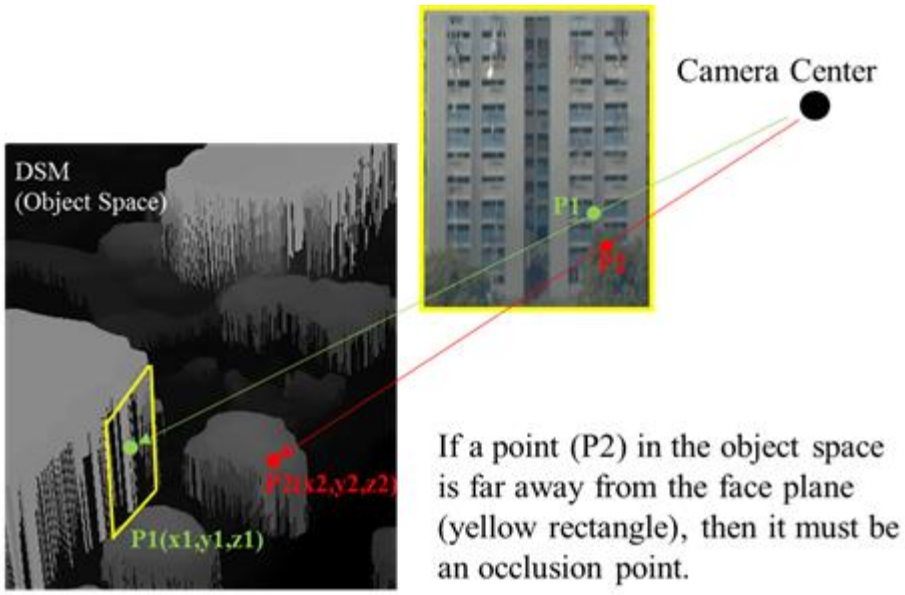



*Figure 5-54. Illustration of the occlusion detection through Z-buffer with the DSM.*

For each face region in the multi-view oblique images, we can shoot rays from its camera centre through the face points to hit the DSM in the object space. If a pixel is not part of the face (occlusion), like the P2 in Figure 5-54, the position where it hits the DSM will be far away from the face plane. Hence, based on the distance of the hit points and the face plane, we can effectively find the occlusion as the black areas illustrated in the rightmost images of Figure 5-53.

**c) Side View Feature Extraction**

To capture the side view features, average colour and the standard deviation in R, G, B channels are calculated, while the histogram of oriented gradients (HOG) [16] and Haar-like features [17, 18] are adopted for the texture description.
The HOG descriptor counts occurrences of the gradient orientation in different localized portions of an image with a histogram. By normalizing and concatenating all local HOGs, such as different parts of a human body, the object boundary can be efficiently described. In our case, the entire side face is treated as a single portion because there is no dominant or specified distribution in the side views. On the other hand, considering the elements (e.g., windows) in the building faces usually have a regular and repetitive layout, we adopt the rectangle Haar-like features which have been applied to identify the face as another description of the side view. The rectangle Haar-like feature is defined as the difference of the sums of the pixel intensities inside different rectangles. For the side view textures, a three-rectangle Haar-like structure is designed and used at the vertical and horizontal direction, separately, with 3 different sizes (total 6 features). Finally, from pixels to blocks, the colour, gradient and Haar-like features are combined to describe the side view for each above-ground segment.

**d) Classification with Side View and Top View**

Followed the idea of object-based image classification, firstly, we segment the land-cover image into small segments as the basic classification units. Then, for each segment, the top view features are directly extracted from the orthophoto and DSM, while the side view features are assigned based on their overlaps with the segments of above-ground objects. Finally, with the top view and side view features, a random forest classifier is trained for the land-cover classification.

**e) Image Segmentation with Superpixels**

Several image segmentation algorithms have been used for the remote sensing data, such as mean-shift [19, 9] and superpixel segmentation [7, 20]. Without valuing the shape as a main rule, the mean-shift algorithm can generate segments with an object-like shape, but the size of segments varies a lot and the result is sensitive to the algorithm parameters leading to unpredictable segments. Oppositely, the superpixel algorithm generates compact segments with regular shapes and scales, which are more robust and suitable to attach the side view features without unexpected mistakes. Hence, in this study, we generate the SLIC superpixels and assign each superpixel the side view features based on their overlaps with the above-ground segments, as illustrated in Figure 5-55. On the other hand, for the superpixels not in the above-ground areas, their side view features will be set as zeros.

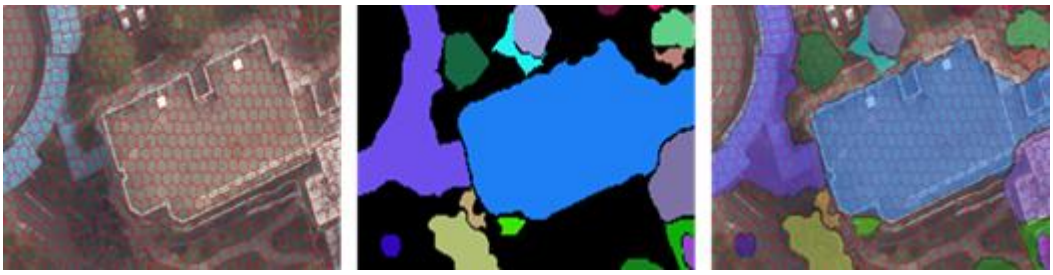


*Figure 5-55. Assignment of side view features for each superpixel. From left to right, the images are the orthophoto with superpixels, above-ground segmentation (colour blocks) and their overlap.*

**f) Classification Workflow**

In this work, we directly adopted the classification framework introduced in [10] which uses a dual morphological top-hat profile (DMTHP) to extract the top view features and the random forest to classify the segments. More specifically, the top view features include the dual DMTHP features extracted from the DSM and brightness and darkness orthophoto images which are produced by the principal component analysis (PCA). The DMTHP extract the spatial features with a class-dependent size which is adaptively estimated by the training data. This mechanism avoids exhaustive morphology computation of a set of sizes with regular intervals and greatly reduces the dimensions of the feature space. On the other hand, the random forest classifier is widely used for hierarchical feature classifications. The voting strategy of multiple decision trees and the hierarchical examination of the feature elements make this method have high accuracy. The entire classification workflow can be found in Figure 5-56 and more details about the top view feature extraction and the random forest classifier can be discovered in [10].

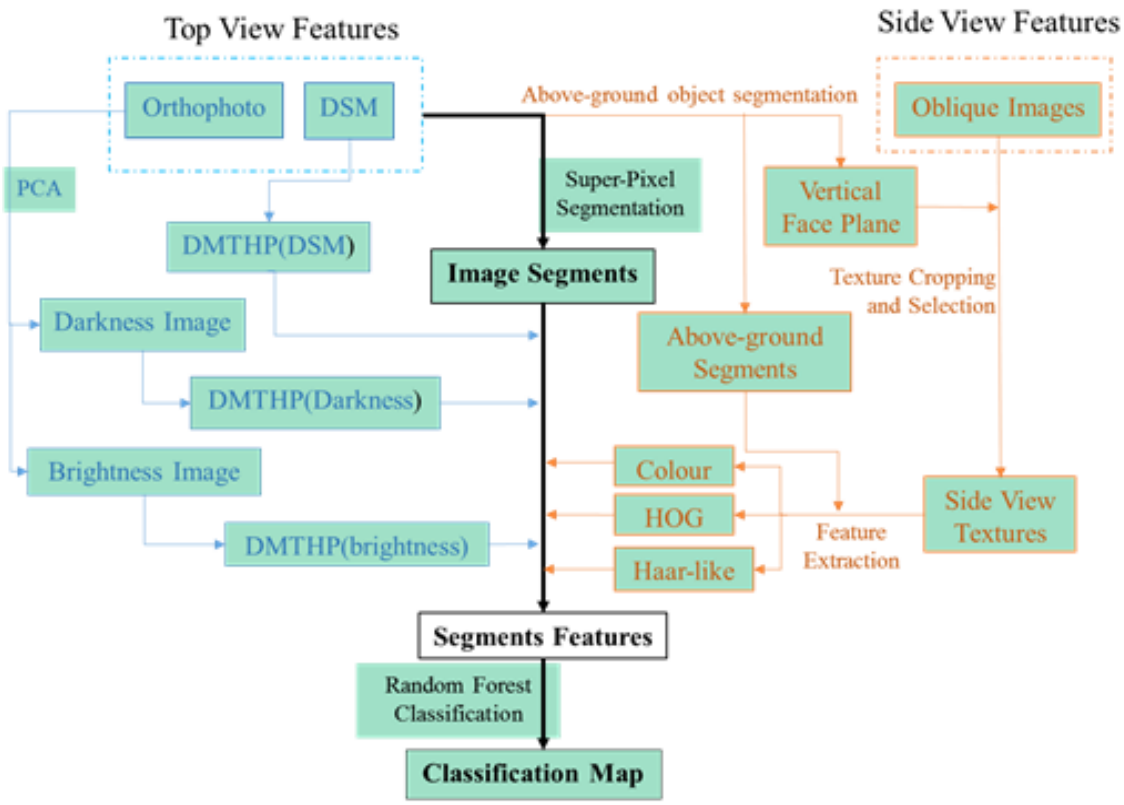



*Figure 5-56. The proposed workflow for the land-cover classification.*

### 5.2.1.3 Experiment

**a) Data Processing and Experimental Setup**

In the experiment, 306 aerial images are used as the study data which includes 73 top-view, 64 forward-view, 47 backwards-view, 62 left-view, and 60 right-view images taken by a 5-head Leica RCD30 airborne camera. The size of all images is 10336 pixels × 7788 pixels while the four oblique cameras are mounted with tilt angles of 35 degrees. These images are georeferenced by a professional photogrammetric software called Pix4D Mapper© which is also used to produce the orthophoto and DSM. The georeferencing accuracy, computed from 9 ground control points, is 0.029 m. The ground sampling distance (GSD) of the orthophoto and DSM is around 7.8 cm.

The study area is around the campus of the National University of Singapore (NUS), where the terrain contains a hilly ridge with tall and low buildings, dense vegetation, roads, and manufacture structures, as illustrated in Figure 5-57. To analyze the improvement of incorporating side view information in the land-cover classification, six sites with different surfaces are selected as marked in Figure 5-57. Site A is a complex campus area which includes dormitories, dining halls, study rooms, and multifunction buildings. Site B and Site E are residential areas with different types of residential buildings. Site C is a secondary school with the education buildings and a playground on a roof, while Site D is a parking site. Site F, with a complicated land-cover, is a much larger area which will be used for a generalization test.

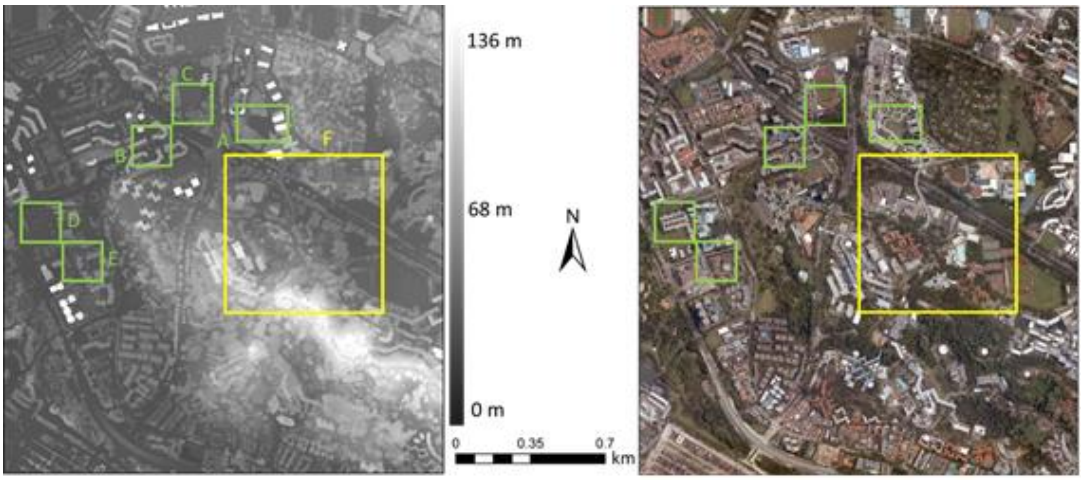



*Figure 5-57. The study area around NUS campus with six experiment sites marked by rectangles.*

Based on the location of this study area, the land-cover is classified into 1) ground including the road, bare ground, and impervious surfaces, 2) grassland, 3) car, 4) tree, 5) rain-shed including pedestrian overpasses, and 6) building. Other objects, such as water and lower vegetation are not considered. The reference masks of the land-cover are manually drawn with a visual inspection of the orthophoto, DSM and the oblique images. For each test site (except site F), around 2% of the labelled segments are used to train the classifier and more statistics about the experimental setup are listed in Table 5-3. For the random forest classifier, 500 decision trees are used for training, while the number of variables for classification is set as the square root of the feature dimension which is 36 in the experiment.

*Table 5-3. Statistics of training and test samples.*

| | Site | A | B | C | D | E |
|---|---|---|---|---|---|---|
| Training sample for each class | Ground | 50 | 51 | 50 | 51 | 50 |
| | Grassland | 50 | 51 | 49 | 51 | 44 |
| | Car | 13 | 53 | 7 | 52 | 41 |
| | Rain-shed | 49 | 52 | 49 | 27 | 51 |
| | Tree | 50 | 51 | 50 | 50 | 51 |
| | building | 51 | 50 | 51 | 50 | 50 |
| Total training samples | | 263 | 308 | 256 | 281 | 287 |
| Total test samples | | 14791 | 7095 | 7883 | 8099 | 6006 |
| Total segments | | 17285 | 11230 | 11237 | 11232 | 11236 |
| Percentage | | 1.52% | 2.74% | 2.28% | 2.50% | 2.55% |

### b) Validation of Above-ground Object Segmentation

The above-ground object segmentation is an initial and critical step for the side-view information extraction. To validate the above-ground segments, we compared the segments with the reference labels which contain tree, rain-shed, and building. The segmentation accuracy of the

five test sites is shown in Table 5-4. The evaluation metrics includes the class accuracy, overall accuracy, and commission error, corresponding to the percentage of correctly identified above-ground pixels in each class, in total, and the miss-classified above-ground pixels, respectively.

*Table 5-4. The accuracy of above-ground segmentation.*

| Site | Class accuracy (%) | | | Overall accuracy (%) | Commission error (%) |
|---|---|---|---|---|---|
| | Rain-shed | Tree | Building | | |
| A | 97.82 | 83.25 | 96.16 | 93.44 | 1.32 |
| B | 94.64 | 85.02 | 99.14 | 94.55 | 2.92 |
| C | 89.05 | 94.66 | 99.64 | 96.79 | 0.39 |
| D | 99.92 | 63.17 | 98.81 | 85.33 | 4.91 |
| E | 79.42 | 84.80 | 98.40 | 93.71 | 2.54 |
| Avg. | 92.17 | 82.18 | 98.43 | 92.76 | 2.42 |

As observed from Table 5-4, most of the above-ground pixels have been successfully segmented out (92.7 % overall accuracy) and in which only a few are miss-classified (2.42 % commission error). For class accuracy, most of the buildings are identified correctly, but with some fuzzy edges. This is mainly caused by the smoothing operation in the generation of DSM and this operation also reduced the accuracy of the rain-shed which is low and close to buildings. On the other hand, some pixels of trees are not identified mainly due to the limited 3D reconstruction of the three branches, as illustrated by rectangles in Figure 5-58. In addition, different objects may be segmented as one if they are close and have similar heights. This kind of error may make objects have wrong side-views like the rain-shed would have the side-views of trees, as marked by the circles in Figure 5-58. However, this error will not severely damage the final classification, because the side-view is just a complementary information, the top-view features still plays an important role in the final classification.

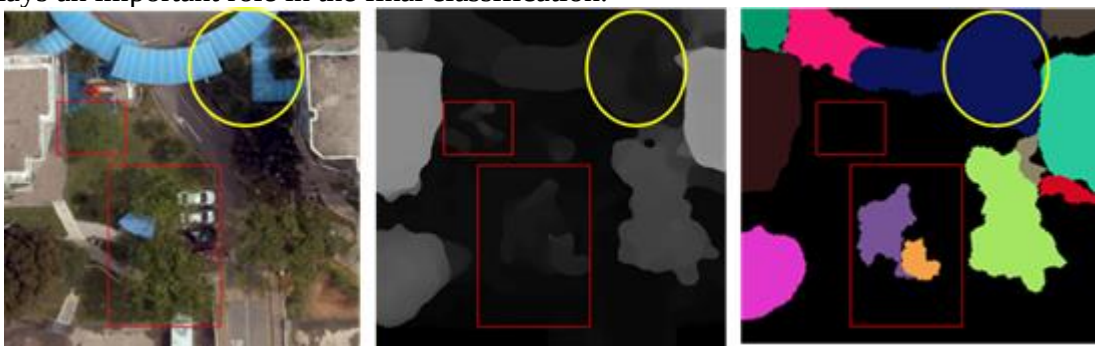


*Figure 5-58. Illustration of the errors in above-ground object segmentation. From left to right are the orthophoto, DSM, and above-ground segments with different colours. Rectangles mark the missed trees while the circles mark a wrong segment which contains multi-objects.*

### c) Classification

For supervised land-cover classification, the training samples are critical. In practice, there are two ways to collect training samples: 1) Evenly collect samples over the entire test site which we refer to as global samples; 2) Selectively collect samples covering part of the test site which we refer to as local samples. As they are illustrated in Figure 5-59, the global samples can offer abundant intra/inter-class information, but they need a considerable amount of labor with scrutiny over the entire image. On the other hand, using local samples can reduce the manual work and is more efficient at larger scales, but they may be not enough to have the generality of

the entire categories. Considering that these two samples concepts are both very common in practice, we experiment with them, separately.

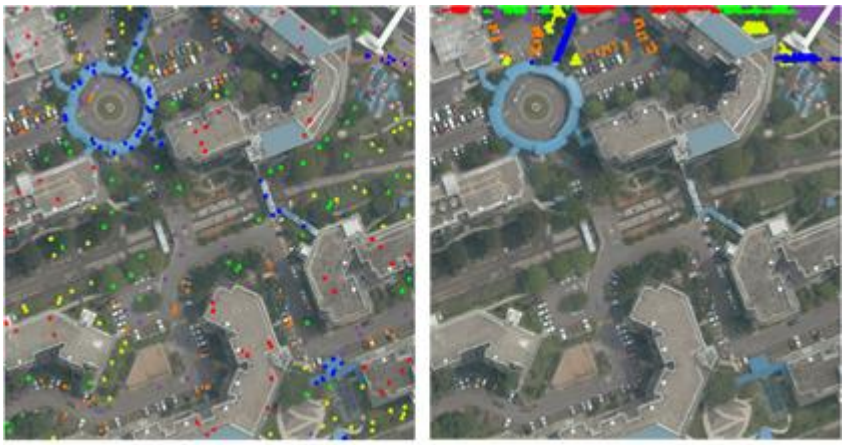



*Figure 5-59. Training samples with different collection methods. The colours represent different classes.*

### d) Classification with Global Samples

The global training samples are selected by evenly picking up the superpixels with certain intervals based on the ratio of training samples in the entire reference data. Following the training and prediction process as described in previous section, we performed the classification with/without the side view features, separately, and the results are given in Table 5-5, where the user accuracy is measured.

*Table 5-5. land-cover classification user accuracy (%) with global samples.*

| Site | Side View | Ground | Grassland | Car | Rain-shed | Tree | Building | Overall Accuracy | Kappa |
|---|---|---|---|---|---|---|---|---|---|
| A | Yes | 93.52 | 95.70 | 89.36 | 93.67 | 94.11 | 90.63 | 92.65 | 89.46 |
| | No | 92.40 | 95.93 | 92.25 | 92.64 | 87.30 | 89.12 | 90.83 | 86.85 |
| B | Yes | 88.01 | 85.54 | 87.08 | 93.15 | 93.12 | 96.42 | 92.06 | 89.52 |
| | No | 85.34 | 86.95 | 87.04 | 91.58 | 92.09 | 96.42 | 91.22 | 88.44 |
| C | Yes | 92.74 | 94.61 | 73.73 | 96.33 | 96.96 | 99.45 | 97.26 | 95.86 |
| | No | 92.66 | 94.62 | 73.73 | 95.70 | 97.48 | 99.12 | 97.30 | 95.93 |
| D | Yes | 93.04 | 95.07 | 78.42 | 90.24 | 89.95 | 97.27 | 92.73 | 90.64 |
| | No | 92.14 | 94.43 | 79.42 | 88.65 | 86.20 | 90.87 | 90.45 | 87.67 |
| E | Yes | 95.08 | 95.78 | 81.50 | 88.17 | 92.71 | 96.30 | 95.15 | 92.37 |
| | No | 94.83 | 95.14 | 81.02 | 88.17 | 91.87 | 95.87 | 94.71 | 91.68 |
| Avg. | Yes | 92.58 | 93.31 | 82.02 | 92.31 | 93.36 | 95.97 | 93.96 | 91.56 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| No | 91.47 | 93.41 | 82.69 | 91.35 | 90.99 | 94.28 | 92.90 | 90.11 |

As we can observe from Table 5-5, the results with the side view have higher overall accuracy and kappa values (in the average improved by 1% and 1.4%) which means the side view information offers useful clues for the land-cover classification. If we do not consider the objects (ground, grassland, and car) which are not benefited from the side view information, the average overall accuracy improvement is 1.7 %, especially for the trees which we will discuss more in the example of site A, as illustrated in Figure 5-60.

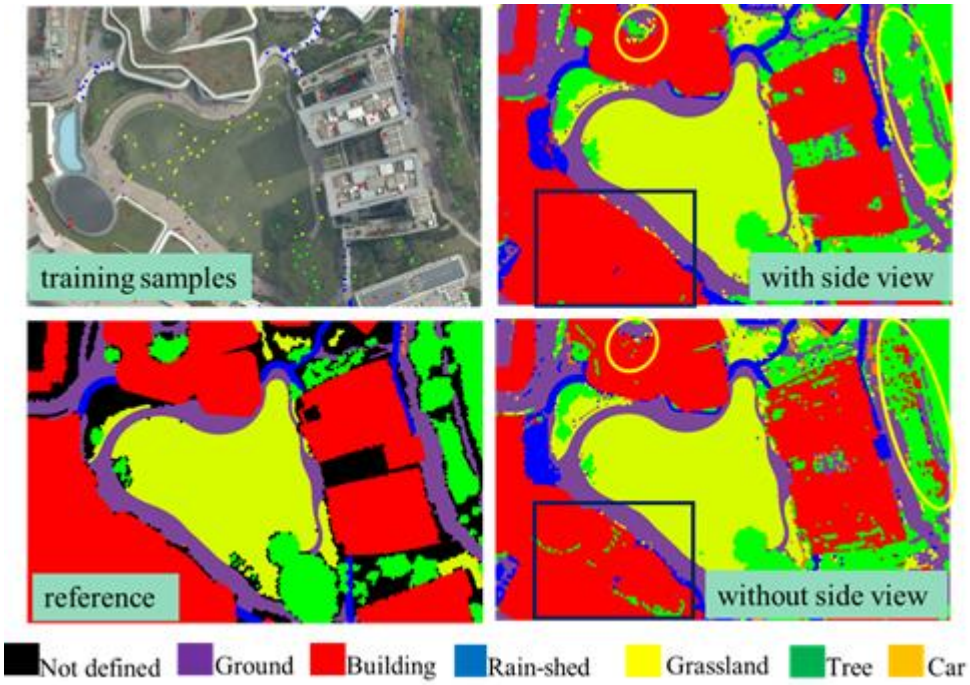



*Figure 5-60. Classification results in site A.*

As we can observe from Figure 5-60, the classification without side view incorrectly classified some trees into buildings (marked by circles). This error is mainly because, in this region, many buildings have vegetation-covered roofs which make their top view features have higher similarity to the trees. One the other hand, some tropical trees with dense and flat crowns, could have very similar top view features compared to building roofs covered by vegetation. Also, low vegetation on the roof, as marked by the rectangle in Figure 5-60, could be misclassified as trees since they have enough height. However, with the differences in their side views, for example, trees are usually more green and darker in the side views, the classifier could identify fewer trees as building, and vice versa.

Somehow, there could be errors in the side view information incorporated classification, as it was marked by a circle in Figure 5-61, where the trees are better identified without side views. From the 3D visualization of this area, we can observe that the trees are growing through a hole of a building roof which makes the trees become a part of the building and share a building side view. This kind of error is mainly from the incorrect above-ground segments. However, even the superpixels of trees are assigned with a building side view, their top view still insured some of them are correctly classified.

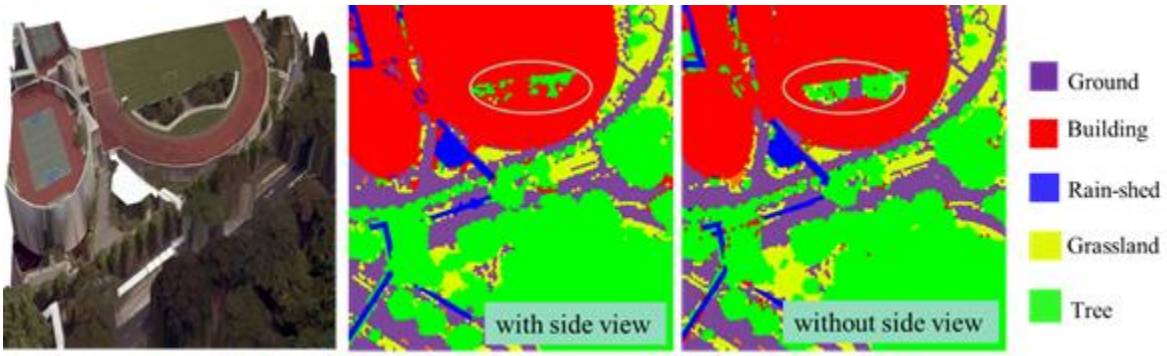



*Figure 5-61. Classification results in site C.*

### e) Classification with Local Samples

As illustrated in Figure 5-59, the local training samples are generated by directly selecting the first portion of the labelled data. With the same training and prediction process, the classification user accuracy with/without side view features are given in Table 5-6.

*Table 5-6. The land-cover classification user accuracy (%) with local training samples.*

| Site | Side View | Ground | Grassland | Car | Rain-shed | Tree | Building | Overall Accuracy | Kappa |
|---|---|---|---|---|---|---|---|---|---|
| A | Yes | 37.97 | 97.08 | 95.27 | 70.15 | 70.23 | 94.20 | 82.88 | 74.72 |
| | No | 36.11 | 97.55 | 95.20 | 62.49 | 74.26 | 58.91 | 65.94 | 53.79 |
| B | Yes | 68.13 | 86.32 | 68.75 | 82.10 | 74.07 | 97.75 | 82.59 | 77.31 |
| | No | 70.23 | 82.95 | 65.59 | 80.81 | 88.85 | 84.50 | 80.36 | 74.44 |
| C | Yes | 62.65 | 87.23 | 92.06 | 79.57 | 97.80 | 97.69 | 92.00 | 87.99 |
| | No | 65.04 | 85.26 | 92.06 | 80.69 | 89.03 | 97.85 | 88.86 | 83.23 |
| D | Yes | 58.26 | 58.42 | 47.80 | 99.97 | 71.90 | 97.15 | 67.76 | 60.76 |
| | No | 57.69 | 57.01 | 39.70 | 99.97 | 67.82 | 97.15 | 65.87 | 58.39 |
| E | Yes | 80.75 | 78.32 | 77.90 | 97.33 | 82.50 | 93.99 | 87.09 | 81.61 |
| | No | 76.46 | 86.78 | 77.21 | 94.35 | 82.33 | 93.23 | 85.65 | 79.62 |
| Avg. | Yes | 61.65 | 81.96 | 76.53 | 84.65 | 77.26 | 96.23 | 82.23 | 76.08 |
| | No | 61.11 | 81.91 | 73.95 | 83.66 | 80.46 | 86.33 | 77.34 | 69.89 |

As compared to the results of global training samples, the local samples have a degraded performance (more than 10% lower for both overall accuracy and Kappa). As mentioned, the local samples can only offer the object-level information which is not enough to represent the entire category. However, in such a situation, the side view still improved the average overall accuracy by 5%, especially for the building which has been improved by 10%. However, the side

view information has reduced the accuracy of trees by 3%. This is mainly due to the improper training samples that contain noisy side views. As we observed, the trees close to buildings, as well as the rain-shed, could have unstable side view features due to the occlusion, even it has been considered in the feature extraction. Thus, if with only limited training samples, this instability may damage the training leading to unreliable predictions. Nevertheless, with high quality training samples, even limited, the involving of side view can still greatly improve the land-cover classification as demonstrated by the classification of the buildings.

**f) Generalization Ability of Side View Information**

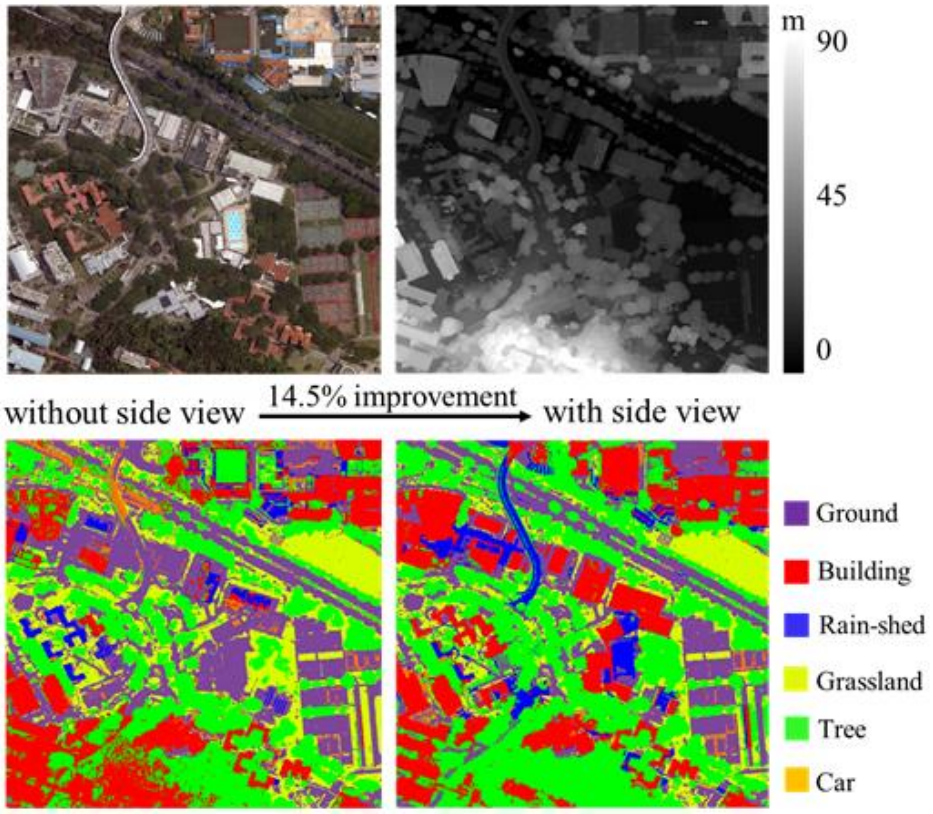



*Figure 5-62. Land-cover classification at the center of NUS campus.*

To further analyze the generalization ability of the side view information, we experiment with the trained classifier at a much larger area (site F which is 16 times larger than other sites) at the centre of NUS campus. Due to the very high resolution of this data (GDS is 7.8cm) which contains 8262*8721 pixels, we downsampled it to one-third of the original size. In this area, a total of 180152 superpixels/segments are generated, containing short and high buildings, tropical trees with smooth canopies, interchanging roads and constructions along the ridge of a hill. In the experiment, the classifier is trained by the reference data of previously mentioned five test sites, and the classification results (with/without side view) are shown in Table 5-7 and Figure 5-62.

*Table 5-7. User accuracy (%) of the classification in site F.*

| With Side View | Ground | Grassland | Car | Rain-shed | Tree | Building | Overall Accuracy | Kappa |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Yes | 86.45 | 76.86 | 8.45 | 40.06 | 95.31 | 85.61 | 87.04 | 81.96 |
| No | 87.96 | 85.71 | 0.79 | 5.53 | 79.74 | 57.24 | 72.54 | 63.04 |

From the results, we can observe that the side view has greatly improved the classification accuracy. With the side view, the overall accuracy and Kappa value have been improved by 14.5% and 18.9%. For the above-ground objects, the overall accuracy of the building and rain-shed have been improved by 28.37% and 34.53 leading to a large average improvement of 26.2% (including the tree). Without side view features, many buildings are identified as ground and rain-shed, while trees are identified as buildings. This site is a very complicated area with various man-made objects and dense trees crossing a hill with large topographic relief (more than 90 meters). In Singapore, many buildings with green/sports roofs could be confused with trees, grassland, and playground from the top view. Especially, if the buildings are surrounded by trees or at the hillside, they could be classified as ground since their elevation features are damaged by the relief of the DSM, as shown in Figure 5-63. Unlike the top view or the elevation features that could have various variations or vulnerable to the DSM relief, the side view features are much more consistent and robust to the change of scenarios. As Figure 5-63 illustrated, with the side view information, the buildings at the hillside are correctly classified, as well as the one with a playground roof.

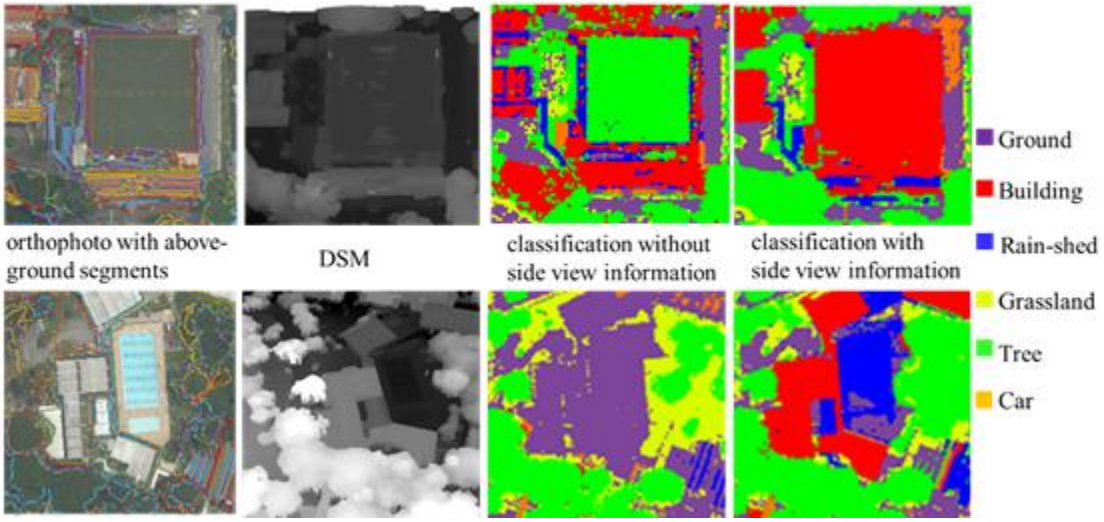



*Figure 5-63. Classifications of complicated areas.*

Somehow, there are still some misclassified areas even with the side view features, for example, many rain-sheds are not classified correctly. There are two main challenges for rain-shed identification: 1) The rain-sheds are short and in many cases, they are close to the buildings and trees which will make their elevation features and side views not clear; 2) There is no significant feature in the side views of the rain-sheds because they are totally random in most cases. On the other hand, we found the ground objects, such as ground and grassland, have slightly worse performances with the side view. This degradation is mainly caused by the errors in the above-ground segmentation. Many ground areas are wrongly segmented as above-ground objects due to the limited precision of the DSM that many ground areas surrounded by trees or buildings can be

mixed with these high objects. Also, cars are barely classified because we have downsampled the data by three times which damaged the superpixels of cars.

As mentioned above, the classification could be sensitive to the training samples. To analyze this sensibility, we changed the training data by alternately removing the samples generated from the previous five test sites. To better analyse the effects of incorporating the side view, we separately trained the classifier with/without the side view features while the training samples are removed site by site. The final classification accuracy and Kappa values, as well as their average (Avg.) and standard deviation (Sd.), can be found in Table 5-8.

*Table 5-8. Effect of changing the training samples.*

| Without Samples From | | A | B | C | D | E | Avg. | Sd. |
|---|---|---|---|---|---|---|---|---|
| With Side View | Overall Accuracy | 85.28 | 86.4 | 86.13 | 87.13 | 85.65 | 86.12 | 0.71 |
| | Kappa | 79.68 | 81.06 | 80.65 | 82.13 | 80.02 | 80.71 | 0.96 |
| Without Side View | Overall Accuracy | 70.51 | 77.0 | 72.4 | 70.93 | 71.54 | 72.48 | 2.63 |
| | Kappa | 60.78 | 68.77 | 62.97 | 61.0 | 61.77 | 63.06 | 3.31 |

As we can observe, when the training data changes, the classification with side view is more stable, indicated by the smaller standard deviations for both overall accuracy (0.71 vs 2.63) and Kappa (0.96 vs 3.31). From Table 5-8, we can find the training samples from site A and D are important for the top view features, as the classifications have obviously decreased without their training samples. However, with the side view information, the importance of these samples is reduced. Especially the side view features even become better without the samples from site D where there is a parking lot. In general, as we compared, the classifications with side view always have better performance indicating the side view features are more steady and robust to the randomness of the training samples.

## 5.2.2 Facade Semantic Segmentation

### 5.2.2.1 Introduction

The facade is an important element in the 3D reconstruction of the city model. , A detailed building façade with semantic labels can greatly benefit many computer vision applications, such as the street map reconstruction and automatic driving, movies and 3D games, and architectural design. Also, the semantic labels of a building façade can be used for the simulation of building energy analysis, façade painting and clearing cost estimation, and building function classification.

However, the building façade semantic labeling is still facing many challenges, such as the variations of different environments, the illumination change, and occlusions. In this research, we propose an efficient way to combine the image segmentation and the deep learning based convolutional neural network (CNN) for the façade semantic labeling.

### 5.2.2.2 Orthographic Mean-shift for Image Segmentation

We observed that most elements in the façade, such as windows, pilasters, and balconies, are both orthographically distributed which means they have strong geometrical constraints at both vertical and horizontal direction. Also, for most façades, their elements usually have a lot of

repetitions in the two orthographic directions which makes that the façade has many regions sharing the same appearance. Hence, based on this observation, we propose to use a mean-shift based method to segment the façade into different elements or regions that can be further analyzed for semantic labeling.

**(a) Façade Rectification**

Since the algorithm is depended on the façade elements repetitions of horizontal and vertical directions, the façade images must be rectified, as Figure 5-64 shows below.

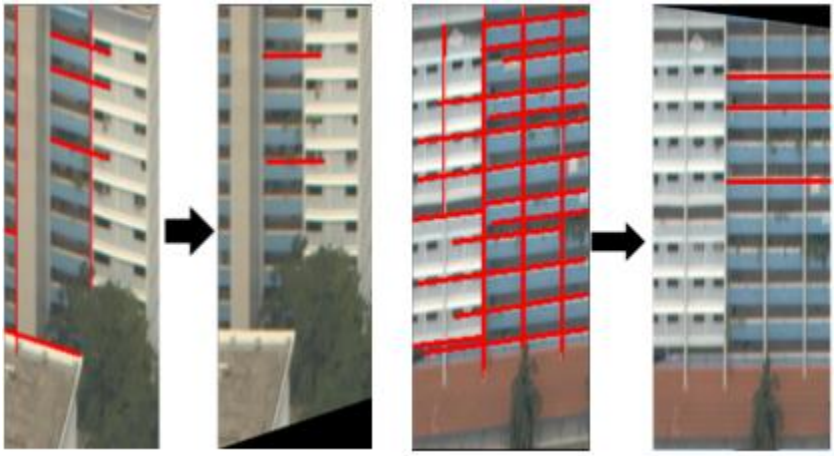


*Figure 5-64. Orthographic rectification of façade images.*

Usually, the façade images are first cropped from the oblique images based on the model or their 3D locations. However, due to the incorrect or limited 3D location accuracy, the façade image may not be orthographically placed. Hence, to correct it, we first detect lines by Hough transform and then use a transform to make them at the horizontal level, such as illuminated in Figure 5-64. Firstly, we find all detected lines' slopes and use a histogram to analyze its main distortions in both horizontal and vertical direction. Then, by force the two main directions to be orthographic, we applied the homography transform on the whole image.
This method usually works well when the façade elements are regularly distributed. However, when the elements are not clear or staggered, the detection of façade lines will be distorted and this will further affect the rectification.

**b) Orthographic Mean-shift Clustering and Segmentation**

With this rectified image we can perform the spatially-irrelevant mean-shift clustering to segment the image. As we can observe form most façade images, from top to bottom, many image rows are very similar. Hence, based on this, we can treat each row as a feature vector and cluster all the rows into different groups. For example, for an image with size 125*100 pixels, we can convert each row into a sample with 100 feature dimensions. Then, the regular mean-shift clustering can be applied to the total of 125 raw samples. Like the use of image compression, the 125 raw samples can be represented only by their group centers which could be much less than 125. Similarly, for the repetitive columns we can also perform the clustering to reduce the representative columns.
Figure 5-65 gives an indication of the spatially-irrelevant orthographic clustering and segmentation. By selecting a good mean-shift clustering band and merging the isolated centers

we can smooth and represent the image with limited row and column samples, just as the right images shown in Figure 5-65. Consider both the horizontal and vertical clusters, the whole image can be finally segmented into different rectilinear grids as illuminated in the left image of Figure 5-65.

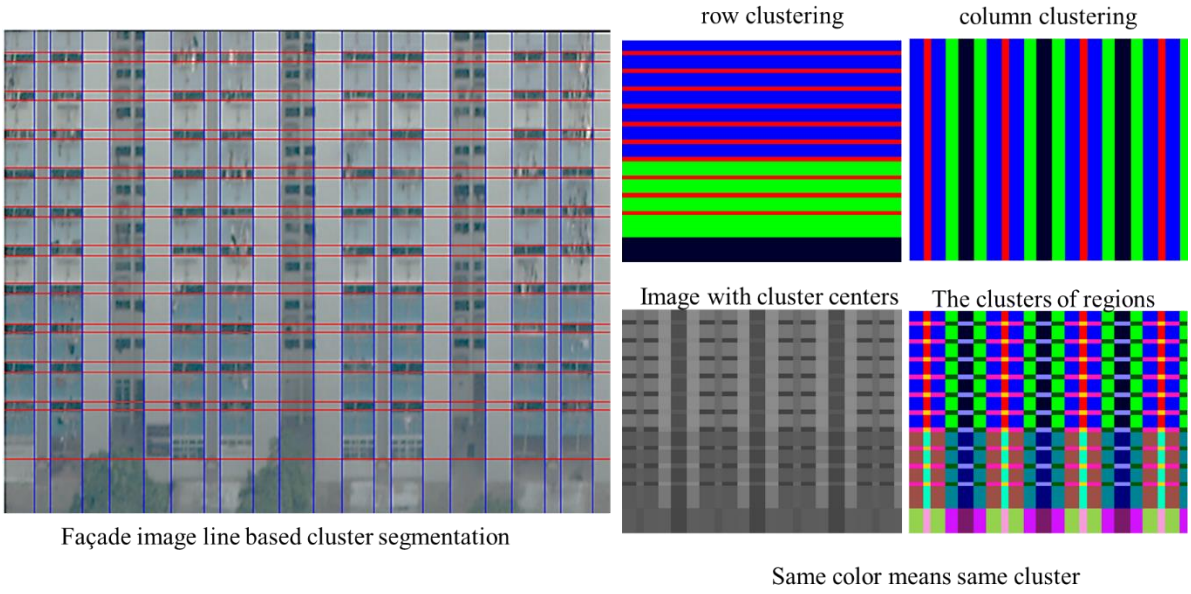



*Figure 5-65. Spatially-irrelevant image orthographic clustering and segmentation.*

As we can observe from the row and column clustering results in Figure 5-65, there are obvious repetitiveness which indicate many grids are very similar and can be divided into same façade category. Thus, we can further group the grids based on their row and column cluster results, as the different color grids shown in the right bottom image of Figure 5-65.

**c) Façade Element Annotation Tool (Based on The Segmentation Results)**

With these façade grids or segments, we can easily and quickly label them and generate the façade semantic labeling training dataset for deep learning.

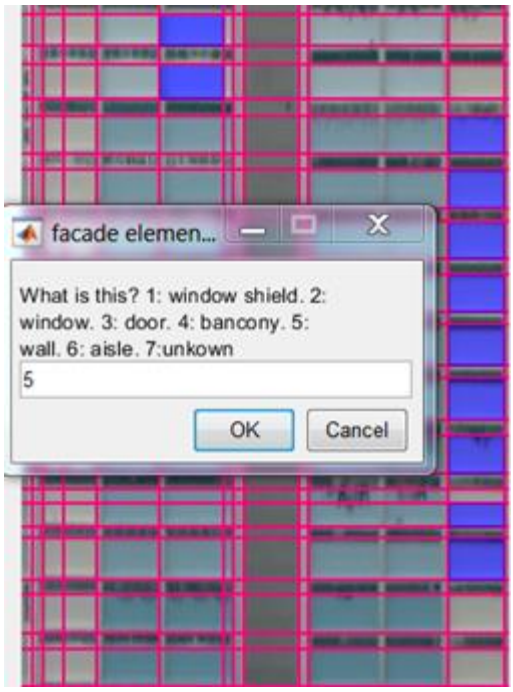



*Figure 5-66. Selection of façade grids and assignment of labels.*

**Labeling steps:**

1. By selecting (left mouse click) a grid in the image, the other grids that belong to the same group will be automatically marked.
2. Double click can neutralize the selection.
3. After the selection, by clicking the right mouse button, the labeling window will come out to ask for an input.
4. After the labeling, the selected girds will display with different colors.
5. Repeat 1-4 to get all labels assigned.

In this project, we have cropped and rectified 106 façade images. With the labeling tools, we can efficiently label their façade elements for either semantic segmentation or for the CNN training. Figure 5-67 shows some partly labeled results and we can observe that this tool can efficiently offer us the regular façade element labeling. There may be some label errors because we have idealized all façade elements as regular rectangle grids. However, this kind of error can be minimized by adjusting the segmentation parameters, such as width-band in the mean-shift clustering, to generate smaller segments leading less miss-segmentations.

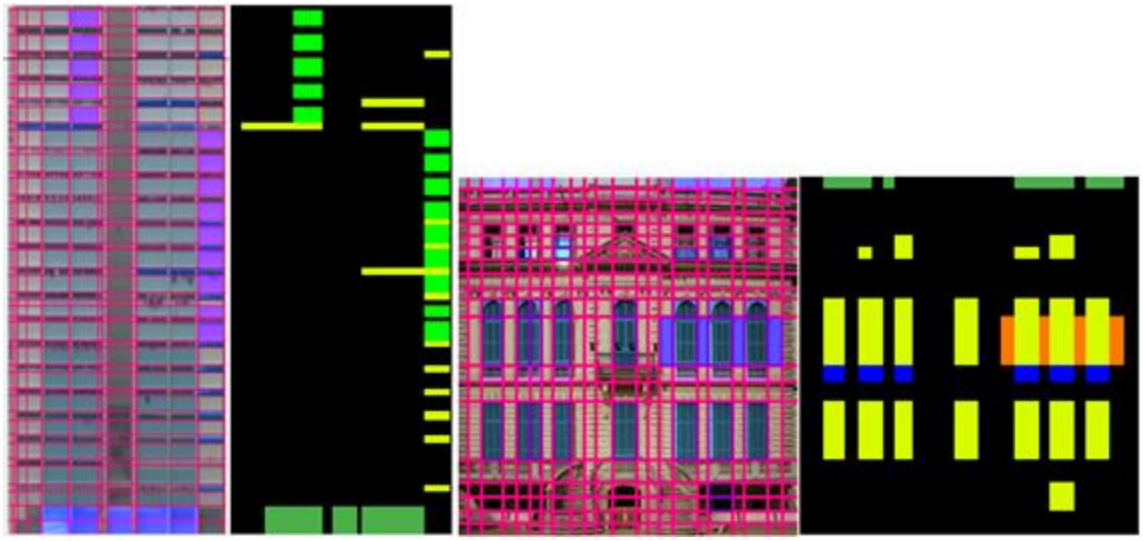


*Figure 5-67. Two examples of partly labelled façade images with grids. Different colors mean different façade elements which are assumed as rectangles.*

**d) Deep Learning for Semantic Labeling**

Deep learning convolutional neural network (CNN) is a powerful tool for some computer vision tasks and it has shown good performance in many applications. In this work, we use a U-Net [21]to classify the façade with different labels. We used 76 images as the training data, while the other 30 images are used for testing. Since U-Net predicts per-pixel labels the yielded segmentations can be blurry at its element boundaries (as shown in Figure 5-68, semantic segmentation image). The segmentations performed using our row/column mean-shift clustering segmentation can further regularize the predicted labels (Figure 5-68, clustered grid image), thus producing a much clearer semantically labelled image (façade element parsing results).

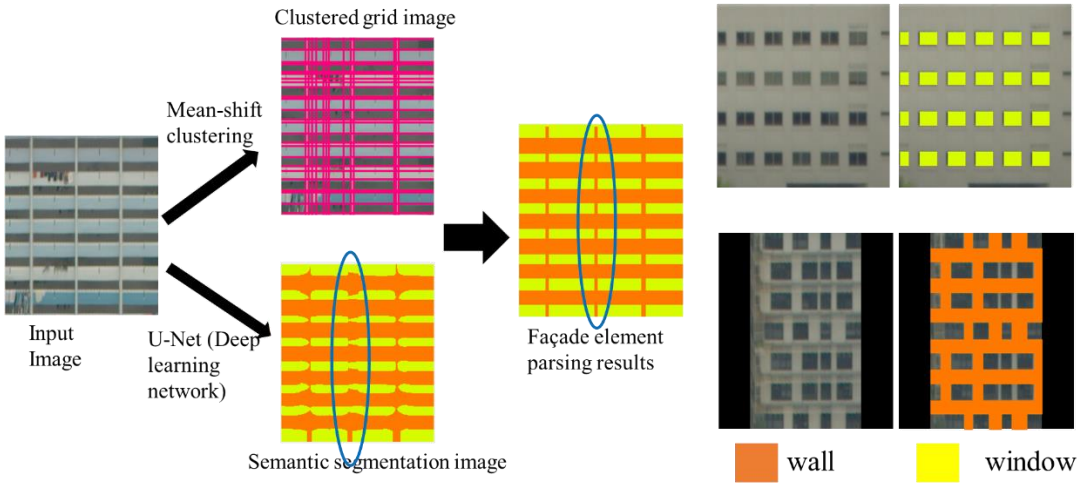



*Figure 5-68. Façade semantic labeling using segmentation and U-Net. Left images give an illustration of the combination of U-Net and Mean-shift clustering results, while right images show some combined results.*

### 5.2.3 Texture Mapping

#### 5.2.3.1 Introduction

Texture mapping refers to mapping 2D texture images onto 3D object surfaces which are usually defined as polygons. Traditionally, building textures are mapped by projecting their 3D polygons into 2D images, finding the corresponding 2D texture and 3D object coordinates with photogrammetric constraints. However, this method has been challenged with the development of the building modelling. By repetitive pattern recognition and procedural modelling techniques, the building model with Level-of-detail-3 (LoD3) is becoming popular and their façades can have hundreds element polygons which need hundreds of texture patches. The texturing of these large amount of façade elements with traditional photogrammetry projection is tedious and error-prone. Since these façade elements are usually generated regularly with some measured procedural parameters, even if they are close to the realities, their geometric structures are not truly measured in the real images, leading to mismatching of the corresponding 2D texture and 3D object coordinates. Façade image segmentation and labeling have been studied for years. Many methods have been proposed and successfully segmented the images into different façade elements. This also inspired us to propose a novel texture mapping method based on the image segmentation. Instead of using traditional pixel-level 3D to 2D projection, we propose to directly match façade elements textures at the element-level without rigorous geometric constraints. By matching the elements in LoD3 façade mode and the segmented segments in the façade image, we can directly get the textures for each façade 3D polygon without the repetitive 3D projection. This method can also resolve the texture occlusion which is usually considered as a serious problem in the conventional texture mapping. With the semantic labels, elements with occlusions can be easily recovered by using other textures that belong to the same category. Additionally, the textures in the same category can be enhanced and unified leading to regular, clear and realistic LoD3 building models.

#### 5.2.3.2 Texture Mapping Based on 3D Projection (for regular façade texture mapping)

3D projection refers to mapping three-dimensional objects to a two-dimensional plane. For the texture mapping purpose, we need to project the façade polygons into oblique aerial images to find their corresponding image textures. Traditionally, the texture mapping includes several steps: 1) 3D polygon projection, 2) texture selection, 3) texture rectification, as the workflow shown in Figure 5-69.

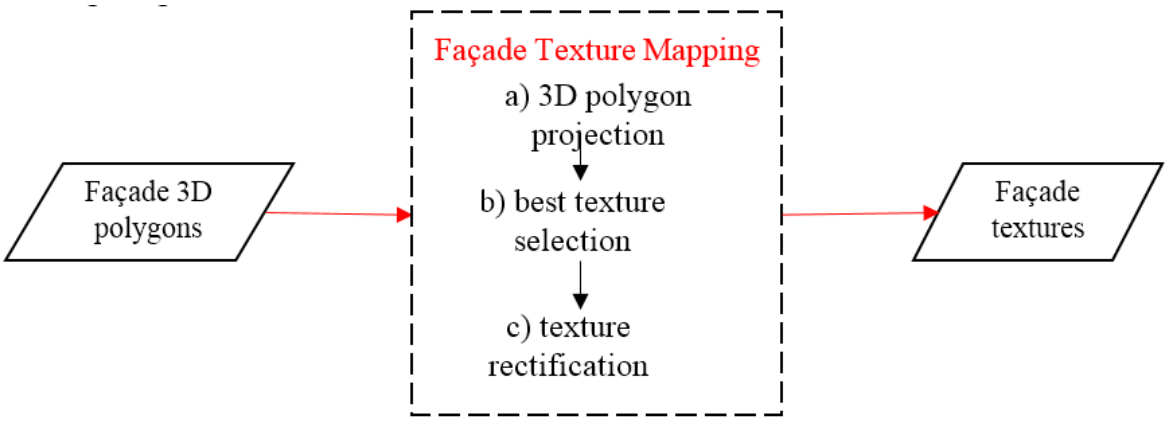



*Figure 5-69. Workflow of the façade texture mapping.*

**(a) 3D Polygon Projection and Façade Texture Selection**

Similar to the side view texture cropping and selection described in section 5.2.1.2, the facade polygons are firstly projected into multi-view image coordinates for the texture cropping. Then, the best textures are selected by a façade texture measurement equation which can be found in section 5.2.1.2.

**b) Façade Rectification**

To better display and analyze the façade textures, the façade images must be rectified as described in section 5.2.2.2.

**c) Texture Mapping Based on Image Segmentation (for LoD3 Façade Texture Mapping)**

After we get the LoD3 building model geometry, it is usually hard to get the façade textures using the 3D projection method as introduced in section 5.2.2.2. The façade model of LoD3 is usually generated by procedural processing which may not precisely match the geometries of the real building, leading to miss-matching after the 3D projection. Instead of projection, we can directly map the textures at the element-level. By semantic segmentation, we can divide the façade images into different segments, representing different façade elements. By matching these façade 3D elements and 2D image segments, the façade texture of LoD3 model can be efficiently mapped without geometric 3D projections, as Figure 5-70 illustrates.

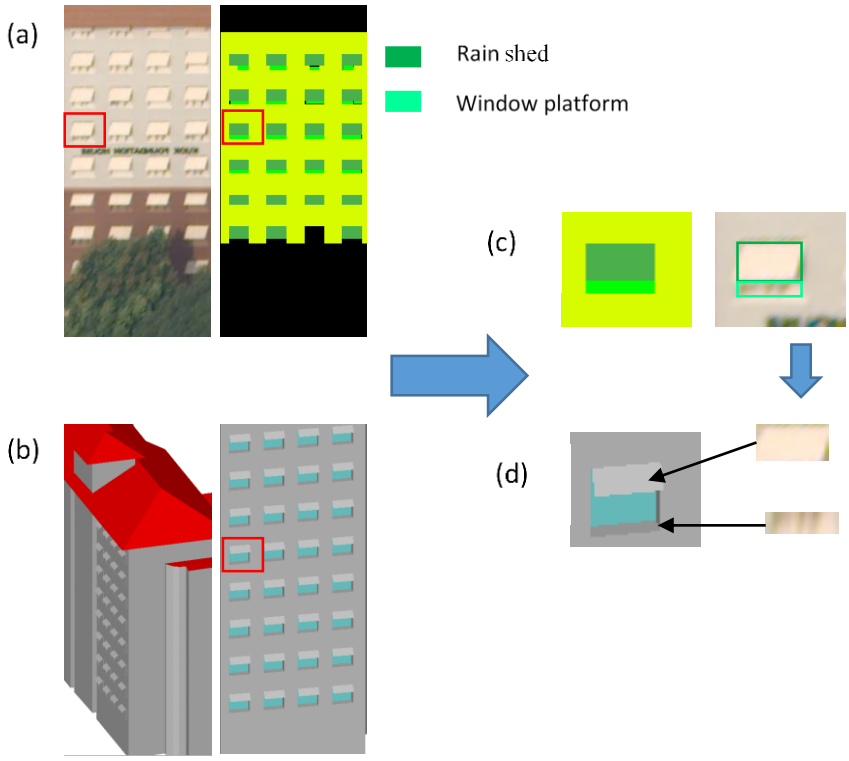



*Figure 5-70. Texture mapping of LoD3 model with image semantic labels. Image (a) shows the semantic segmentation results, while image (b) gives the LoD3 façade model. By matching the semantic labels (image (c)) and façade element (image (d)), the texture of façade model can be mapped.*

**d) Element-level Texture Mapping**

Based on the image segmentation and LoD3 façade model generation, the façade elements can have a one-to-one correspondence in 2D image and 3D object space, as shown in Figure 5-71. With this kind of texture mapping, we don't need to calculate the mathematical 3D projections with complex equations. Façade elements in the 3D model can efficiently find their own photo-realistic textures making the rendering close to the real lively.
However, this kind of texture mapping can come across some problems. The first common problem is the occlusion which makes the modeled façade elements do not have corresponding visible regions in the façade image, as the red rectangle illustrated in Figure 5-71. Another issue will be the occlusion cased miss matching. For example, the window in the LoD3 model has a layout pattern as 8*4, but the pattern of the segmented image is 6*4. How to match these two patterns is still a challenge.

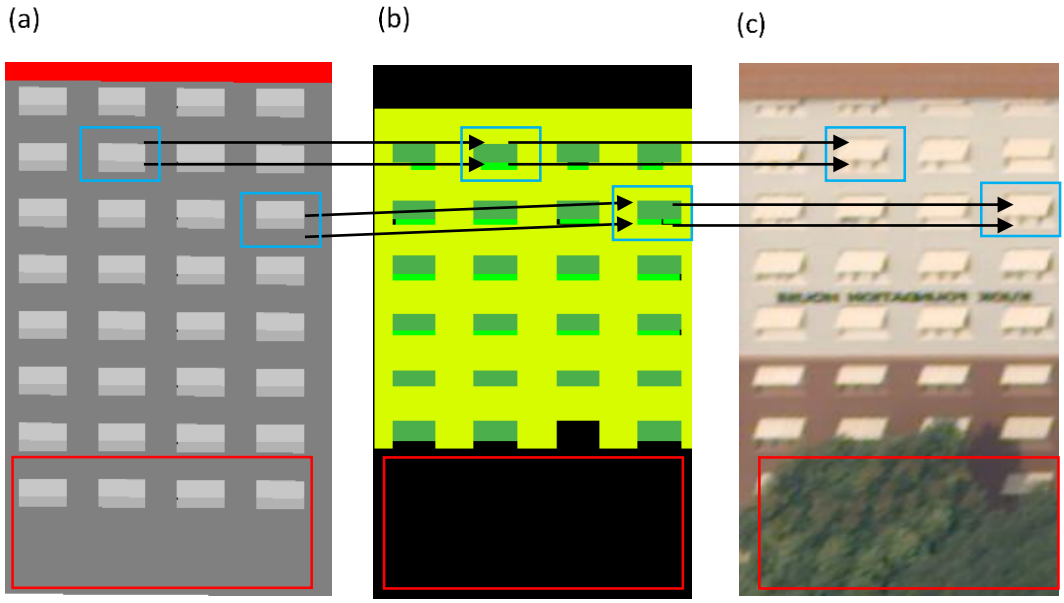



*Figure 5-71. One-to-one texture matches at facade element-level. Based on the LoD3 façade elements (image (a)) and semantic labels (image (b)), each element's texture can be mapped in image (c). Light blue rectangles show two matches while the red rectangle shows an occlusion area without matches.*

**(e) Category-level Texture Mapping**

As we observed, the façade element in the same category usually share the same texture. Hence, in this project, we can use one typical texture to represent all element textures in the same category. There are two ways to select the category representation texture which depends on the texture itself. If the texture is just a pure color block, then the representation texture could be the average color of all the identified image segments in the same category. On the other hand, if the texture contains many patterns, such as painting and strips, the representation texture should be the clearest image segment in the category. Using one presentation texture for all category elements is a convenient and efficient way to do façade texture mapping which can resolve the occlusion and matching problems at the same time.

## 5.3 WP3: Interactive Post-editing and Procedural Modelling of the Building Facades

### 5.3.1 Work Package Introduction and Overview

Since the quality of the models generated by WP1 and WP2 can only be as good as the input data, further refinement of the models may be necessary for certain type of LoD3 applications. An important correction task is e.g. the extension of façade patterns into occluded areas. Due to vertical greenery, which is frequently found on façades in Singapore, underlying façade elements may be covered and therefore not reconstructed correctly by WP1 and WP2. This is why WP3 implements a custom 3D editor that offers *interactive editing and procedural modelling* techniques to enhance the models. Procedural modelling is based on the "Computer Graphics

Architecture" (CGA) shape grammar introduced in [22, 23], which defines a set of operations for transforming shapes in space, with a special focus on architecture. The operations can be combined into rules, which finally define an architectural style.
A set of procedural façade templates is developed as part of WP3 which allows efficient modelling and correction of models. By parameterizing the templates, an open number of façade variations can be generated for numerous application needs (e.g. multiple level-of-detail, schematic vs. textured buildings, combination with zoning laws and planning regulations, or urban design). The editor allows parametrization either through explicit entering of measured values, interactive adjustment via the mouse cursor as well as snapping to the point cloud data. Visually overlying point cloud data with the 3D models in the same editor view enables the operator to adjust the model as well as verify it by comparing it to the actual LIDAR and photogrammetric measurements.
The approach of WP3 integrates the advantages of procedural modelling with reality-based modelling achieved in WP 1 & WP 2, increases the value of the 3D models and extends the range of application of the 3D models.
In addition to that, templates can resolve privacy issues that may arise with high-res imagery by replacing these parts with appropriate geometry and generic textures. Geometry and generic textures can be equally used for street-level modelling where image-based approaches encounter challenges due to the high number of noise (pedestrians, traffic, vegetation, etc.). Thanks to semantic labelling, vegetation can be identified and replaced by procedural plant models for certain applications such as plant grow simulation.
In addition to template based façade modelling, state of the art interactive 3D editing based on push-pull [24] offer the classic polygon editing capabilities and geometric primitives which let the operator handle irregular cases that cannot efficiently represented with rule based approaches (e.g. traditional shop houses in Singapore's Chinatown, see Figure 5-72). Interactive changes are automatically classified and matched to the CityGML data model by building-specific heuristics of the editor. The automatic matching and classification significantly reduce the time required for manual labelling of interactively constructed façade elements.


*Figure 5-72. Traditional shop houses in Singapore's Chinatown.*

### 5.3.2 Interactive Post-editing

#### 5.3.2.1 Introduction

While template-based facade modelling is very effective for modern high-rise, regular architectural styles, varying facades such as those found in Singapore's Chinatown often require considerable manual rework and effective tool support is crucial. Interactive push/pull geometry modifications are state of the art and implemented in most commercial 3D modelling suites (e.g. Trimble SketchUp, Autodesk Fusion 360). They intuitively transform a polygonal mesh locally by interactive vertex, edge or face drag operations. PushPull++[24] adds novel methods for adaptive face insertion, adjacent face updates, edge collapse handling, and an intuitive user interface that automatically proposes useful drag directions. PushPull++ has shown to reduce the complexity for common modelling tasks by up to an order of magnitude when compared to existing tools and is therefore well suited for an efficient LoD3 modelling process. We use a PushPull++ implementation derived from Esri CityEngine for interactive geometry modifications.

#### 5.3.2.2 Interactive Editing Operations

For the Virtual Singapore project, six main types of editings are provided in the 3D editor. They are

1) Mesh optimization (See Figure 5-77)
2) Model move, scale and rotate
3) Vertex, edge and face move (See Figure 5-74)
4) Vertex, edge and face deletion (See Figure 5-73)
5) Face and edge push/pull (See Figure 5-76)
6) Face split (See Figure 5-75)

Use cases for these editing operations are shown below.

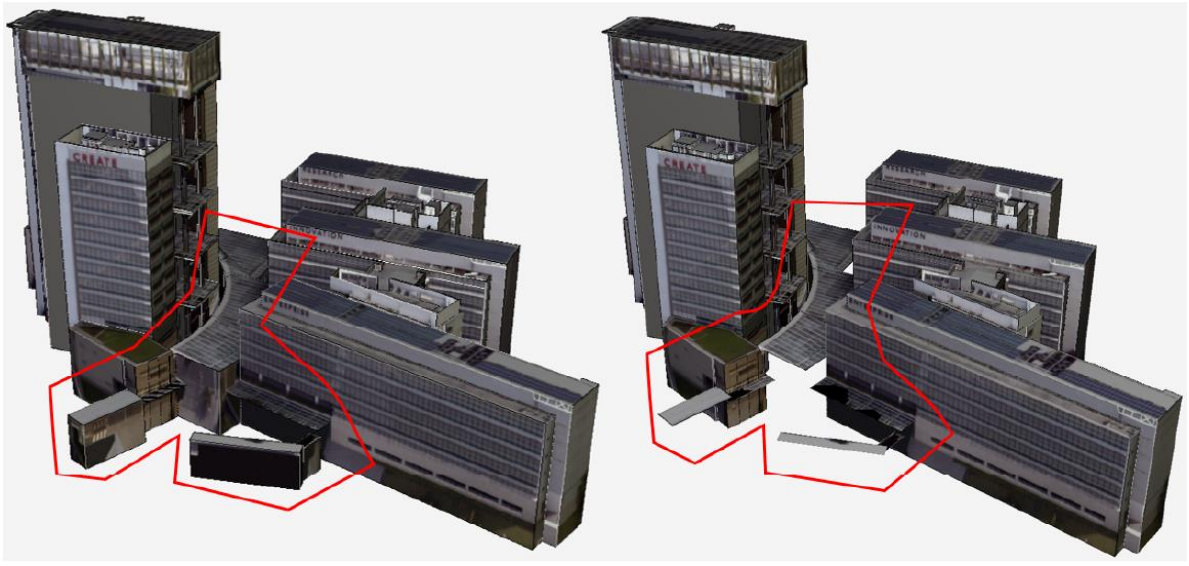


*Figure 5-73. Deletion operation for passageway.*

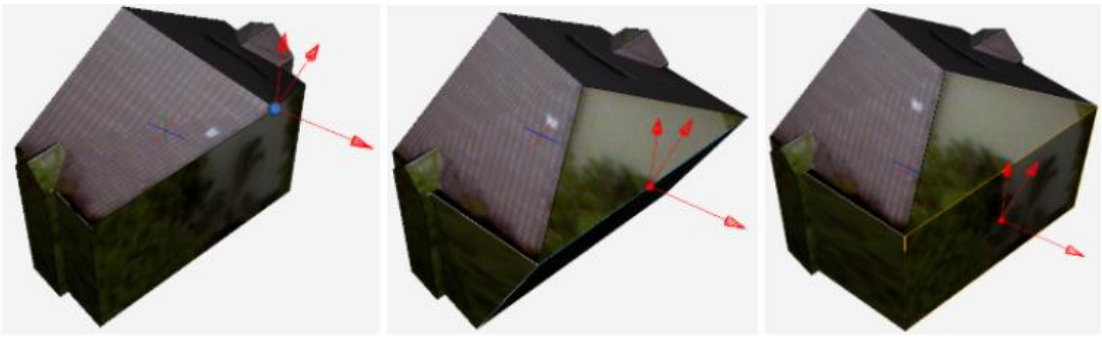


*Figure 5-74. Moving vertex, edge and face.*

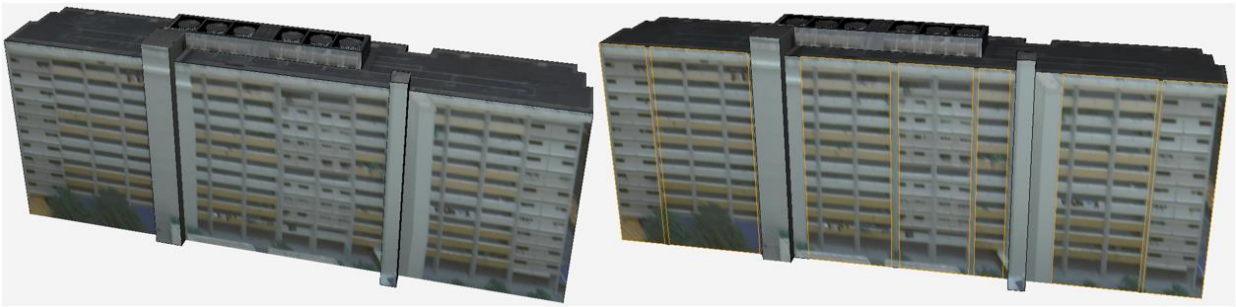


*Figure 5-75. Face split for various facade appearance.*

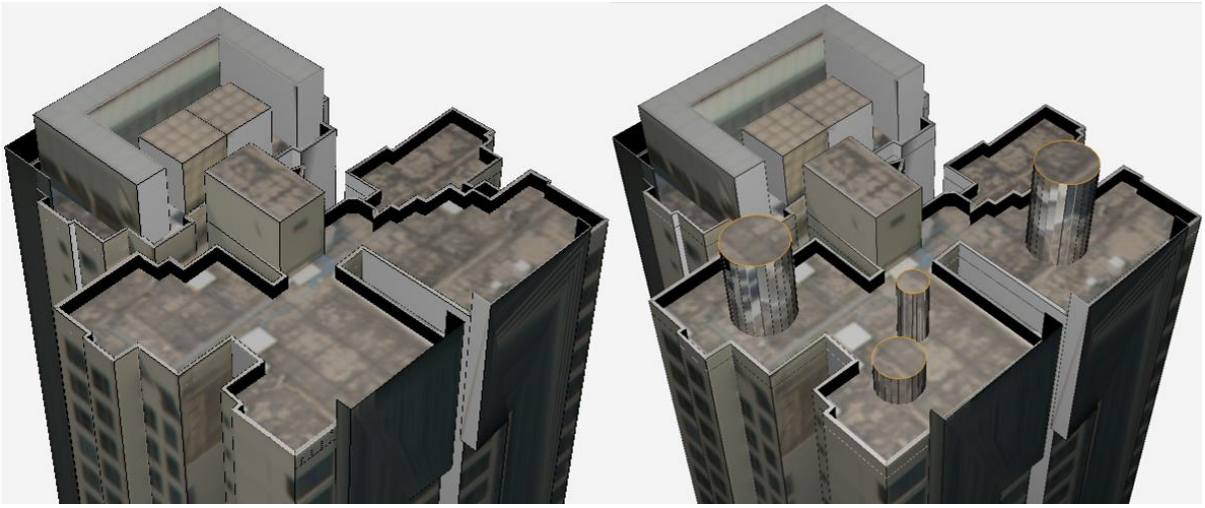


*Figure 5-76. Push-pull for adding roof infrastructure.*

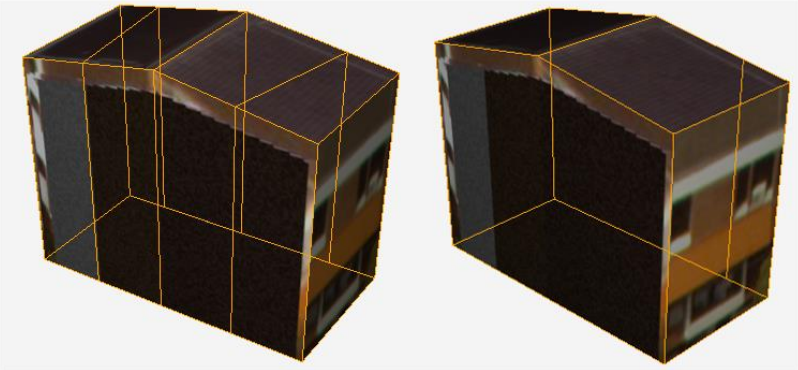


*Figure 5-77. Mesh optimization for simplifying coplanar faces*

In addition to the editing operations, we also support texture for newly added faces as shown in Figure 5-78.

*Figure 5-78. Newly added faces with texture.*

## 5.3.3 Procedural Modelling of Building Facades

### 5.3.3.1 Introduction

Reconstruction based on purely photogrammetric methods faces multiple challenges in an urban environment:

1) Occlusions (e.g. greenery, pedestrians, cars, etc.)
2) Insufficient data due to privacy
3) Duplicate measurement for the same elements

In order to fill these gaps, we are building a reality-based template library which can be parametrized by measurements from WP1 and WP2. Esri's procedural runtime [25] is used to generate geometry from these templates which are then integrated into the final LoD3 building model. The procedural runtime is a library implementing the "Computer Graphics Architecture" (CGA) shape grammar introduced in [22, 23], which defines a set of operations for transforming shapes in space, with a special focus on patterns found in architecture.

### 5.3.3.2 Typical Templates for Singapore's Building

We designed four typical facade templates for the appearance library. They are:

1) Opening template
2) Awning template
3) Carpark template
4) Extrude++ template

As illustrated in Figure 5-79, these facade templates are designed based on characteristic of buildings in Singapore. Openings with awnings are common in Singapore's buildings, facade with carpark is also popular. Figure 5-80 shows the user interface to parameterize the facade template.

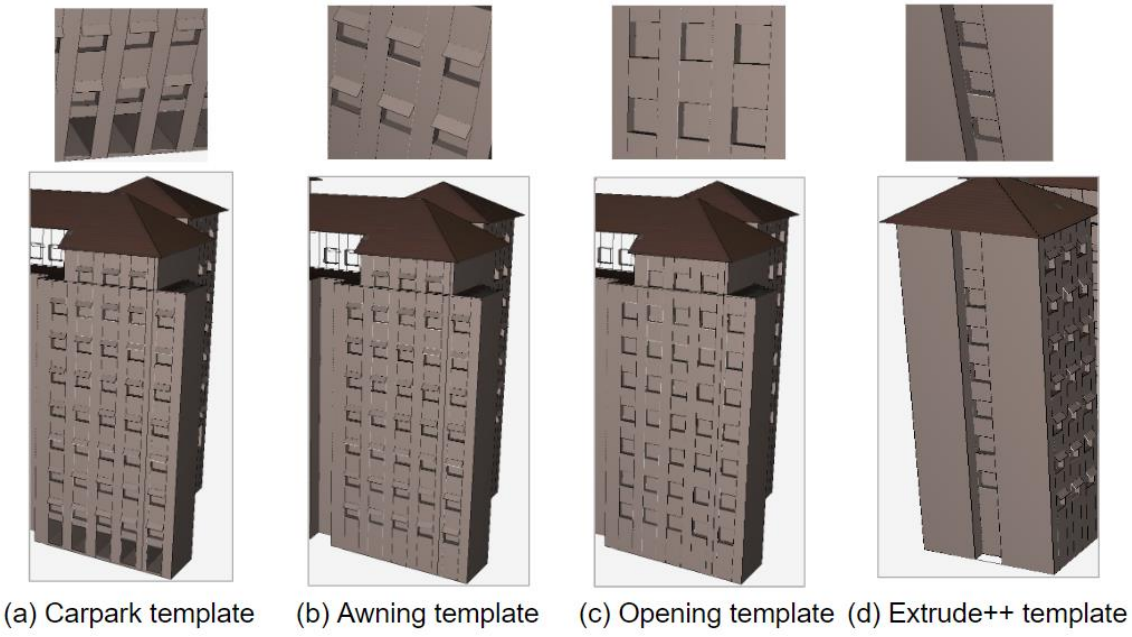



*Figure 5-79. facade templates for appearance library.*

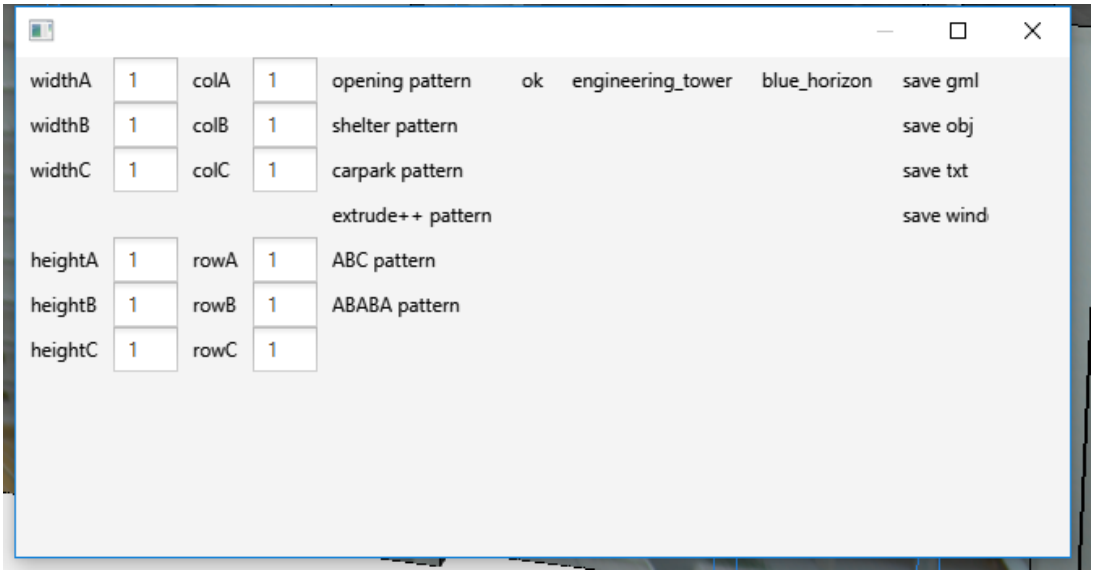



*Figure 5-80. User interface for parameterizing facade templates.*

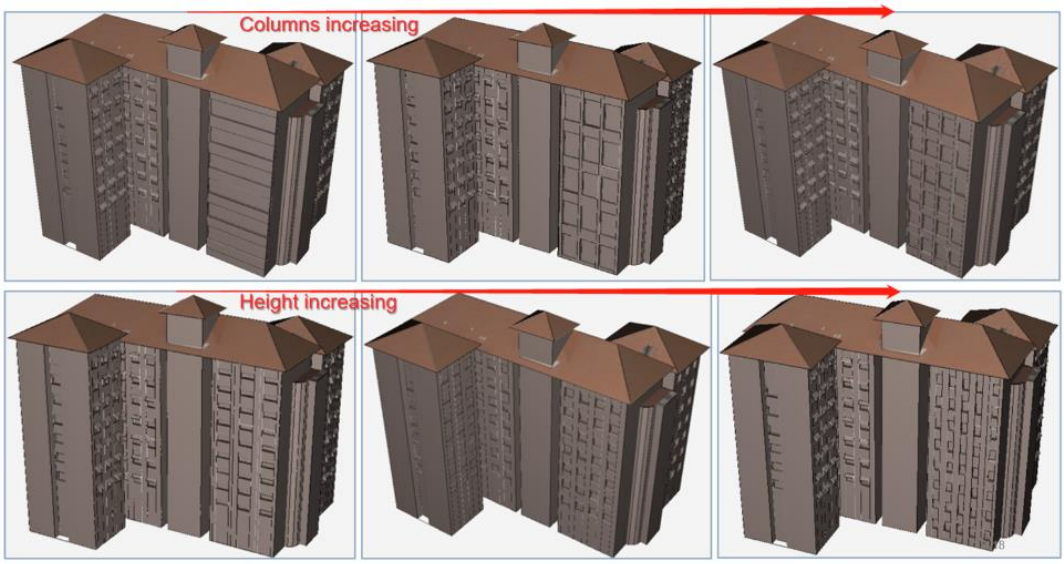



*Figure 5-81. Parameterization of facade templates*

## 5.3.4 Model Matching

### 5.3.4.1 Introduction

Semantic information enables a wide range of application scenarios beyond visualization. Specifically, semantic and hierarchical information of building models is important for building information analysis [26], urban analytic [27], facilities deployment [28], prediction [29] and governing [30]. Semantic information provides the identification of key components in a generic 3D model and enables users to differentiate the faces of wall, roof, ground, and other surfaces of a 3D building model. Hierarchical information carries the relationships between faces in 3D models, for instance, in 3D building models, it tells how a window is related to a wall surface. Nowadays, a variety of techniques are available for editing generic 3D models, however, the maintenance of semantic and hierarchical information through the editing process is only supported by specific applications which are closely tied to the underlying data model. We identified five main classes of popular geometry editing techniques: transformation, deletion, deformation, push-pull, and parametric (procedural) templates. Most editing tools preserve only geometric information, UVs and materials. In order to provide a flexible integration of existing as well as future editing tools, an automatic matching and synchronization method between the modified geometry and the data model is needed.

There are three major challenges in maintaining semantic information across the editing process: the first one is to decide to what extend to depend on the specific edit operation. On the one hand, recovering information based on an edit operation is an intuitive way to maintain semantic information, on the other hand, this would mandate about the same number of data model update strategies as edit operations, severely limiting its flexibility. Therefore, an automatic matching method independent of the specific edit operation is required. The second challenge is how to deal with both rigid and non-rigid transformations of polygons as well as how to distinguish existing faces from newly added faces. New faces need to be inserted and classified according to the data model. The third challenge is the performance of the automatic matching process since it very likely runs after each interactive edit operation and thus may impact the user experience.

The presented automatic matching method addresses all three challenges. It is based on a closed loop of interleaving interactive edit operations and automatic data model synchronizations. Starting from a source data model with geometry, UVs and materials, the method preserves semantic information after geometry changes through a process of registration, recovering, and labelling. The method is independent of the specific editing operation and can thus be applied to a wide range of interactive 3D tools or 3D batch processing jobs. The main contributions of the method are:

1) Independence of edit operation: The method assumes no knowledge about the performed edit operation. It is able to recover information based on the source data model and the modified 3D geometry.
2) Integration flexibility: Since the method is only based on 3D geometry, UVs and materials, which are usually exposed through SDKs and APIs, integration in existing 3D applications is simple.
3) Information preservation: The method can preserve all of the semantic information of the original model, classify newly added polygons, and label them according to the data model.

The method was implemented and tested as part of a LoD3 building editor for the Virtual Singapore project using CityGML as the underlying data model. It features standard 3D transformation tools, push-pull tools, and procedural facade templates through the integration of 3rd party libraries. These libraries disregard the data model and operate on the 3D geometry only. The CityGML data model is updated automatically after each edit operation. 30 editing test cases were designed to verify that the data model is consistent, and the semantic information is maintained after each edit operation.

### 5.3.4.2 Related Work

Several research areas are relevant for understanding the problem context: generating and maintaining hierarchical and semantic information of 3D models (CityGML in particular), 3D model matching methods, and state-of-the-art 3D editing techniques.

#### a) Semantic Information

A semantic-enriched 3D scene is crucial for non-visualization oriented applications. Alegre [31] proposed a probabilistic approach to the semantic interpretation of building facades. Verdie [32] reconstructed a 3D urban scene by classifying and abstracting the original meshes. With the assistance of Markov Random Field, Verdie's method is able to distinguish ground, tree, facade, and roof surfaces. Zhu [33] proposed a point cloud classification method based on multi-level semantic relationships, including point-homogeneity, supervoxel-adjacency and class-knowledge constraints. Furthermore, Wu [34] presented an algorithm to produce hierarchical labelling of an RGB-D scene for robotics. Rook [35] automatically label faces using decision tree but without support for LoD3 labels. All these methods address the broader problem of creating semantic information directly from 3D faces or point cloud data and thus do not take into account prior information available in a rich source data model.

#### b) Model Registration

Model registration between the source geometry and the edited geometry allows recovering the transformation applied by an edit operation. Various methods were developed over the past to fully or partially match geometries and extract transformations between matching parts. Sundar [36] proposed a skeleton based method for comparing and searching 3D models. The method expresses the model by a skeleton graph and uses graph searching techniques to match 3D models. DeepShape [37] extracts high-level shape features to match geometric deformations

of shapes. Shape signature is computed with the method introduced in [36], which makes it possible to measure similarity of different shapes. This method represents the signature of an object as a shape distribution sampled from a shape function measuring global geometric properties of the object. Iterative Closest Point (ICP) is a widely used registration method introduced by Besl and McKay [38] and Zhang [39]. This efficient point set registration method meets well our requirements in terms of interactive edit performance and simple geometry representation. ICP proved to be adequate for the edit operations used in our test cases. If necessary, ICP variants such as [40] and [41] can be integrated. Other alternatives are CPD, a probabilistic method introduced by Myronenko [42] for both rigid and non-rigid point set registration. Ma proposed his vector field consensus algorithm in [43, 44].

### c) Editing Techniques

Push-pull (press-pull or sculpting) is a popular, intuitive editing technique which unifies previously distinct tools such as edge move, face move, face split, and extrude into a single tool. Having a single tool minimizes tool changes and thus improves the efficiency of the editing process. It has been adopted by many commercial products, such as AutoCAD [45], SketchUp [46], or Maya [47]. In particular, PushPull++ [24] used in CityEngine [48] and ArcGIS Pro [49] employs methods for adaptive face insertion, adjacent face updates, edge collapse handling, and an intuitive user interface that automatically proposes useful drag directions. PushPull++ has been verified to be able to reduce the complexity of common modelling tasks by up to an order of magnitude comparing to existing tools and is therefore well suited for editing building models and is integrated as part of the Virtual Singapore LoD3 editor. Esri's procedural runtime [25] is used to generate parametrized geometry from facade templates. The procedural runtime is a library implementing the Computer Graphics Architecture (CGA) shape grammar introduced in [22, 23], which defines a set of operations for transforming shapes in space, with a special focus on patterns found in architecture.

### d) CityGML

While the method is independent of the data model, the Virtual Singapore LoD3 editor is based on CityGML [50], introduced by Kolbe [51] , covering the geometrical, topological, and semantic aspects of 3D city models. CityGML defines the classes and relations for relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical, and appearance properties. There is wide industry support for importing CityGML, (e.g. Bentley Map [52], BS Contact Geo, CityServer3D, CodeSynthesis XSD, FME from Safe Software, and many more) but editing of the data model is supported by only a few (e.g. (3Dis, 2019, CityEditor) and (UVM, 2019, CityGrid)). CityEditor is an extension for Trimble SketchUp. The operations of moving, rotating and scaling a building model are well supported by CityEditor. For editing semantic attributes, CityEditor provides a table import and export and a GUI. However, CityEditor cannot maintain semantic information across editing operations which has to be updated manually.

### e) Information Synchronization

The method considers edit operations as black boxes but while looking at current tools, it helped to distinguish five main types of editing techniques: transformation, deletion, deformation, push-pull, and template application. The following section gives an overview of the method and presents each step. Then, the key tasks of registration, recovery, information transfer between source and edited model, and automatic labelling are discussed. Internally, 3D geometry is represented as face-vertex meshes [53] which incorporates a list of vertices and a set of faces referencing their vertices. Both the source geometry and edited geometry are handled as meshes

($M_s$and $M_e$). Mesh $M_s$is represented by vertices $v \in V$ which are points in three-dimensional space. A vertex $v_i$ may optionally be annotated by a two-dimensional texture coordinate $t_i \in T$. For each pair of connected vertices in $f$, an edge $e = (v_i, v_j)$ is defined. Planar faces $f \in F$ are defined by a list of counter clockwise oriented vertices $(v_1, v_2, \ldots, v_n)$. Opening (holes) are represented as faces referencing their boundary (parent) face.

### 5.3.4.3 Method Overview

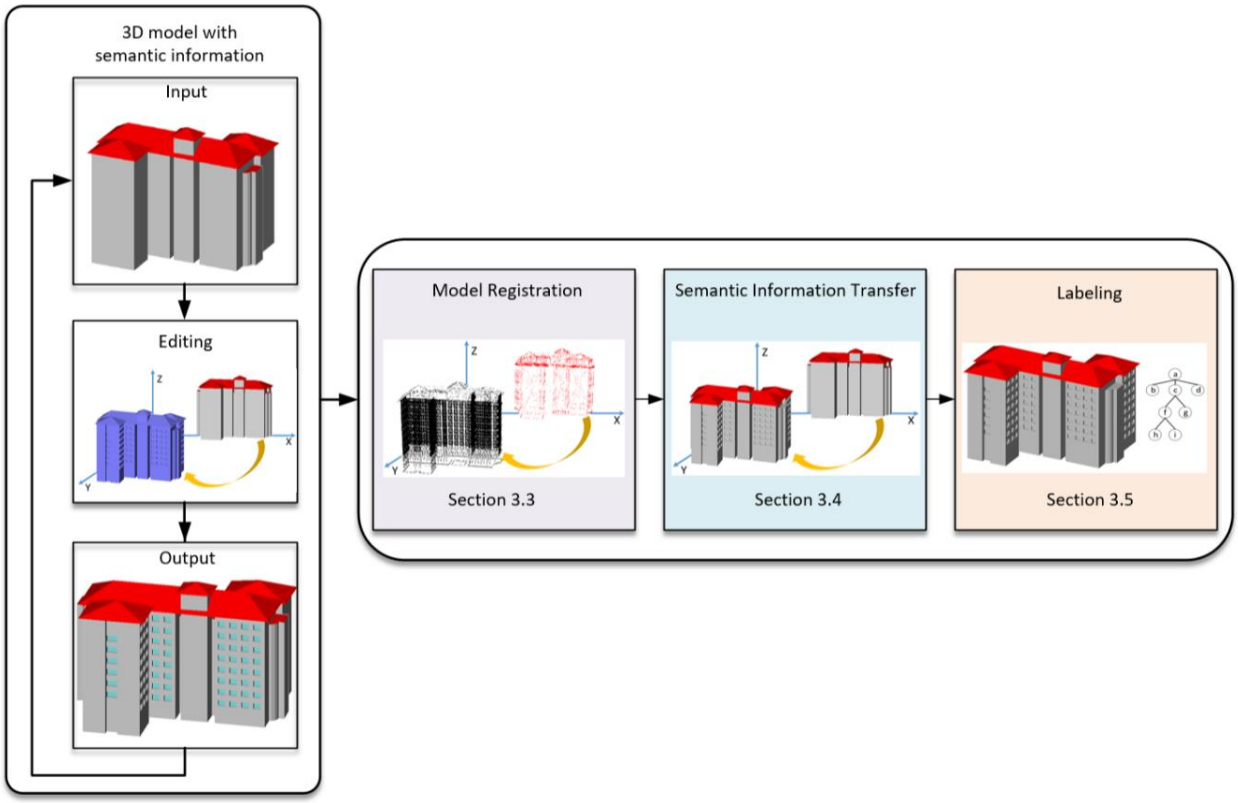



*Figure 5-82. Flow chart of automatic 3D model synchronization*

Figure 5-82 gives an overview of the model synchronization method. Applying an edit operation to a source geometry ($M_s$) results in an edited geometry ($M_e$) deprived of semantic information because we assume that these are not preserved by the edit operation. Is the source point set for the Iterative Closest Point (ICP) algorithm, whereas $M_e$ is its target point set. The output of ICP is a transformation matrix which is applied to the source $M_s$ to yield a transformed point set $M_t$. An octree storing additional information (face normal, face centroid, length of edges) is constructed from the transformed model $M_t$ . An octree search with faces from $M_e$ yields matching rigid transformed faces in and their original semantic labels in $M_s$. Deformed faces are handled in a similar way by an additional UV space search. Each face in the edited geometry $M_e$ is incorporated into a topological tree to detect openings and their boundary surfaces. Boundary surfaces are labeled as roof, wall or ground faces based on their normal direction. For openings, the octree and UV space is used again to find the position in the data model, finally resulting in an updated data model that preserves as much semantic information as possible and which is updated with new faces labeled according to their topology.

### 5.3.4.4 Face Matching

Since $M_e$ contains only geometric information after an edit operation, faces have to be matched with corresponding faces $f_s \in M_s$ to find related semantic information. The following feature are used for face matching:

- Face normal $f_{normal}$
- Face centroid $f_{centroid}$
- Length of edges $f_{length}$
- Set of texture coordinates $f_T$

If the distance between two vertices $v_i$and $v_j$is below a given threshold $\epsilon_v$, they are considered the same. Likewise, if the UV distance of two texture coordinates $t_i$and $t_j$ is below a given threshold $\epsilon_t$ , their UVs are considered the same.

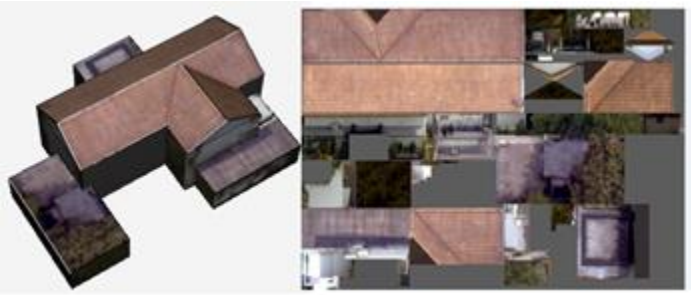


*Figure 5-83. UV space of a building model. In UV space, each vertex $v_i$ of the geometry has a corresponding texture coordinate $t_i$. The UV space simplifies matching faces since it remains unchanged or is re-sampled during deformation, push-pull and template application*

Two faces and are matched if:

- $f_t$ and $f_e$ have the same $f_{normal}$, $f_{centroid}$ , and $f_{length}$
- $f_t$ and $f_e$ share the same set of texture coordinates

For matching faces, semantic information from $f_s \in M_s$ can be transferred to $f_e \in M_e$through $f_t \in M_t$ .

### 5.3.4.5 Model Registration

Several editing operations include rigid transformation and scaling which can be represented by a transformation matrix. The ICP algorithm is used by the method to calculate the transformation matrix between the source and the edited geometry. ICP can be applied to the source geometry as a whole to compute the global transformation of the geometry or partially to recover local transformations.

**a) ICP Algorithm**

The ICP algorithm is a simple and efficient way to find the best match between two point sets by solving an optimization problem. The input of ICP algorithm are two point sets, the source point set *X* and target point set $\begin{matrix} X \\ P \end{matrix} \begin{matrix} = \{x_1, \ldots, x_{N_x}\} \\ = \{p_1, \ldots, p_{N_p}\} \end{matrix}$ (5-9 the point sets may have different size, and the distance between two points is defined by the Euclidean metric.

$$X = \{x_1, \ldots, x_{N_x}\}$$
$$P = \{p_1, \ldots, p_{N_p}\} \quad (5\text{-}9)$$

The ICP algorithm finds the transformation $T$ in an iterative approach, applying transformation $T$ to $X$ and computing the error $E(T)$, until the error is less than a given threshold.

$$E(T) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - T(p_i)||^2$$

**b) Point Set Generation**

*Figure 5-84. Building model with symmetries. Using a uniform sampling strategy will prevent ICP from recovering a rotation by 180°*

In order to increase the quality of the ICP result for a given edit operation, more points than just the vertices of the source geometry are usually required. Generating points by sampling uniformly the edges is easy to implement but will result in the same points for symmetries and thus prevent ICP from recovering e.g. rotation or mirroring along these symmetries (see Figure 5-84). Random sampling is a widely used strategy [54, 55], but may add an error to the ICP result because of the noise introduced by the random sampling. The method uses random sampling along edges and diagonals but with a consistent random sequence based on a random seed computed from the texture coordinate $t_i$ of vertex $v_i$ . As illustrated in Figure 5-85, we generate diagonals by connecting opposite vertices to obtain additional points for the ICP algorithm. If the source geometry does not have texture coordinates, they are generated beforehand. Initializing the random number generator from the texture coordinates results in a unique sequence of sampling points for every edge and diagonal in $M_s$ and hence in $M_e$ , which then can be unambiguously registered by the ICP algorithm.

*Figure 5-85. Edges and diagonals of a face are sampled with a random number generator which is initialized by the texture coordinates of the vertices of the edge or diagonal.*

The edge and diagonal sampling is done as follows:
For a segment $S_{ij}$ with vertices ($v_i$, $v_j$) and texture coordinates ($t_i$, $t_j$),

1) Calculate two random seeds $seed_i$ and $seed_j$ based on the texture coordinates of $t_i$ and $t_j$ .
2) Generate *n* random numbers $r_{i_1}$,$r_{i_2}$,...,$r_{i_n}$ based on $seed_i$, and *n* random numbers based on $r_{j_1}$,$r_{j_2}$,...,$r_{j_n}$. By applying these random numbers, we obtain *n* sampling points $s_{ij_k}$ and *n* sampling points $s_{ji_k}$
3) Add the points obtained by the second step to the source point set and target point set of ICP algorithm

This enables the ICP algorithm to register the source geometry to the edited geometry since there exist unique corresponding point sets for each segment in $M_s$ and $M_e$ .

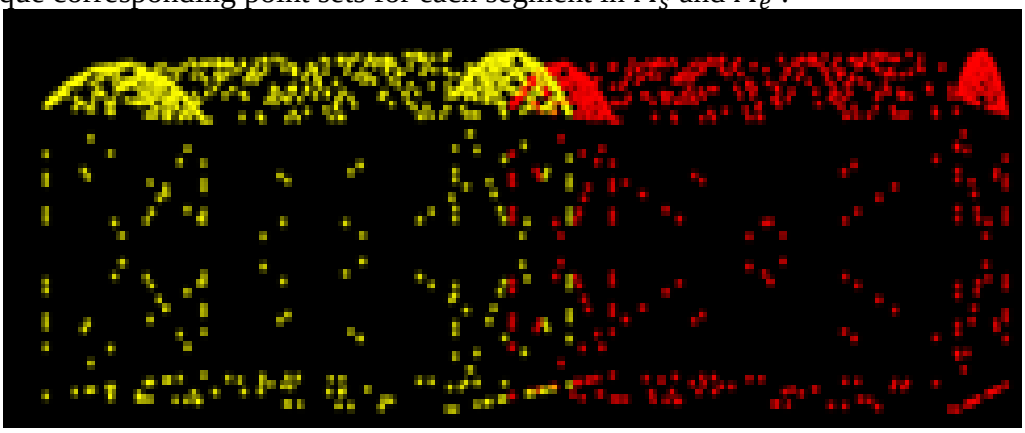


*Figure 5-86. Example of a generated source point (red) and generated target point set for (yellow) for the ICP algorithm.*

**c) Registration Result**

Experiments show that 300 ICP alignment iterations and a distance threshold of 0.01 (the implementation uses meter as unit) are sufficient to calculate the transformation matrix *T*. An example of such a registration result is shown in Figure 5-87. The red point set is generated form $M_s$, the yellow point set is generated from $M_s$ and the white point set is the transformed source point set by applying *T* to source point set.

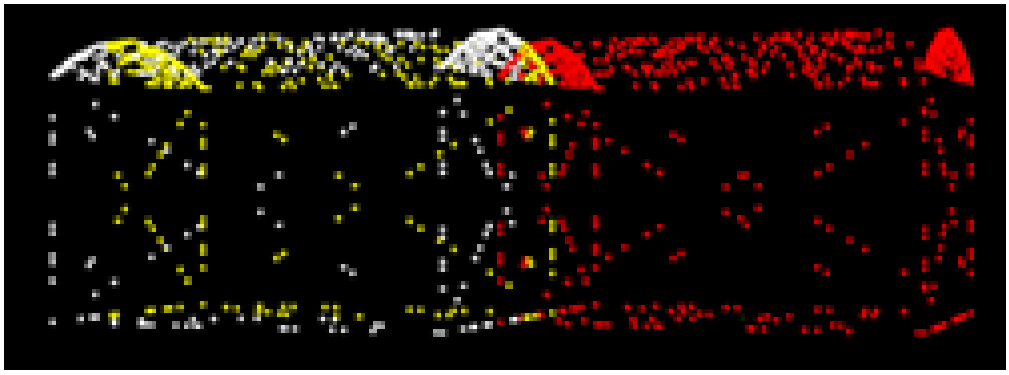


*Figure 5-87. Registration result. The white point set is the transformed source point set. It overlaps with the point set generated from the edited geometry $M_e$.*

### 5.3.4.6 Semantic Information Transfer

To match semantic information from the source model to the edited geometry, a search for matching faces has to be performed. The implementation uses an octree to speed up the search process.

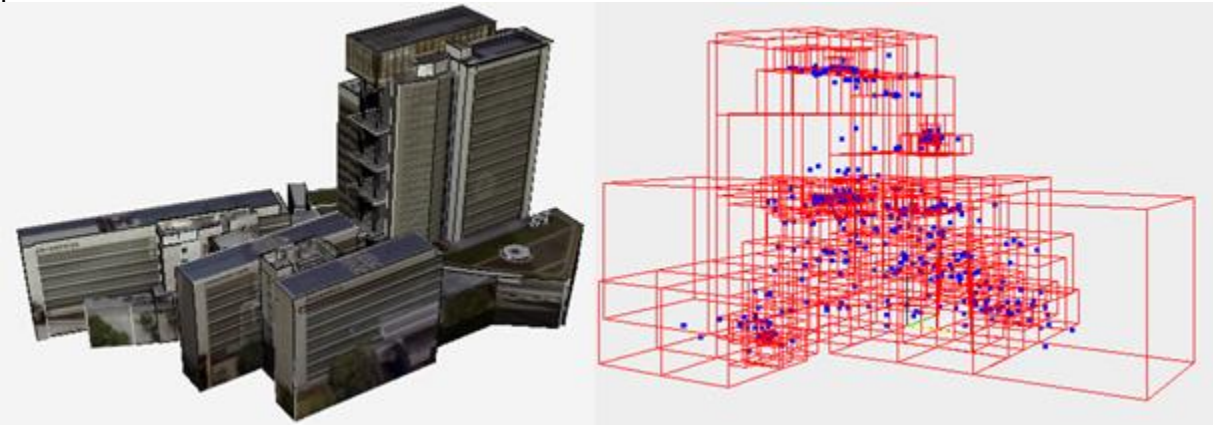


*Figure 5-88. Building model and its octree*

An octree is a hierarchical data structure allowing efficient search operations in three-dimensional space. To match the semantic information of the original model, we construct an octree based on the transformed source geometry $M_e = T \times M_s$ . The root node of the octree is the bounding box of the source model. The bounding box is recursively divided into eight nodes until each box contains only one node as shown in Figure 5-88. Each node stores the face features from the point set generation.

Whether a face $f_e$in the edited geometry matches a face in the source data model is determined by searching for the corresponding face features in the octree. Matching faces allow the transfer of semantic information from the source model $M_s$ to the edited geometry $M_e$.

As illustrated in Figure 5-89, the deformation of a vertex, edge, or face changes $f_{normal}$, $f_{centroid}$, and $f_{length}$. Such a deformed face will not be found by an octree search. However, deforming edit operations only affect the geometry but preserve texture coordinates. Therefore, these faces can still be matched by an UV space search, a method applied also for texture transfer (see for example [56]) and hence be used to transfer semantic information.

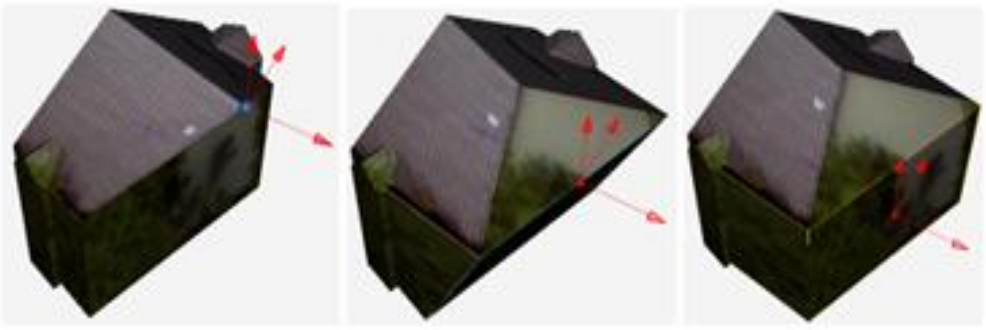


*Figure 5-89. Deformation operations on vertex, edge and face. Deformation will change $f_{normal}$, $f_{centroid}$, and $f_{length}$. A UV space search is performed for these faces.*

After the octree and UV search, unmatched faces in the source data model $M_s$ are deleted and unmatched faces in the edited geometry $M_e$ are considered new faces. For these, semantic information has to be generated through labeling which will be discussed in the next section.

### 5.3.4.7 Labeling

Semantic information transfer based on faces can be applied as described to any data model. However, labeling new faces is domain specific and the approach described here is based CityGML, which is the underlying data model of the implementation. According to the CityGML building model the relevant classes for LoD3 building modelling are divided into three levels (see Figure 5-90), which are building, boundary surface and opening. Specifically, the boundary surfaces are roof surface, wall surface and ground surface, whereas the openings include windows and doors.

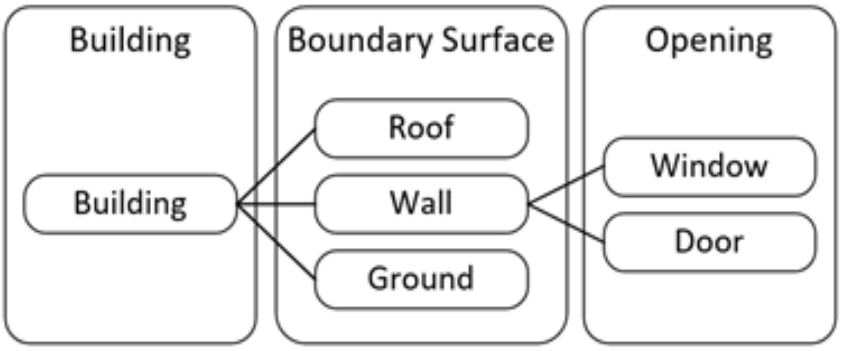



*Figure 5-90. Relevant parts of the CityGML data model for LoD3 modelling.*

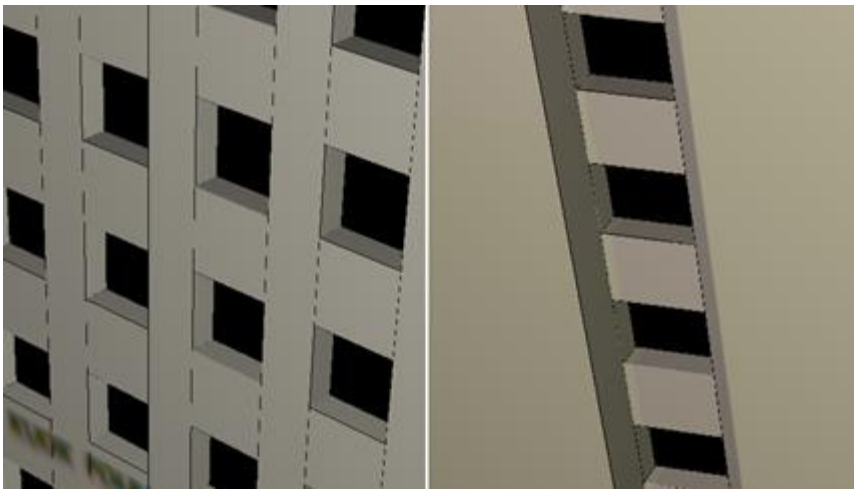


*Figure 5-91. Openings generated by template application. This editing operation will split the boundary surface and generate several new faces co-planar with the boundary surface.*

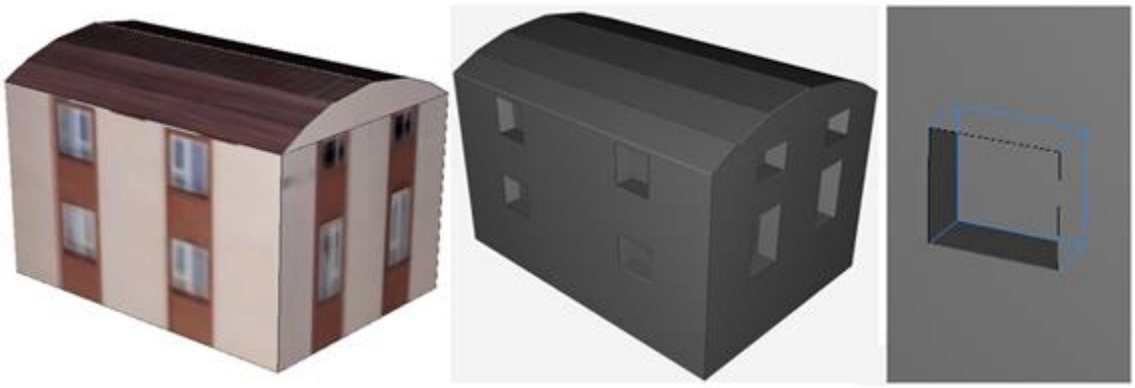


*Figure 5-92. Opening generate by push-pull. This editing operation will not split the boundary surface but generate a hole on the boundary surface.*

Typical edit operations which create new boundary surfaces or openings are push-pull (see Figure 5-92) and template application (see Figure 5-91). New boundary surfaces are classified according to the following heuristic where $f_\phi$ is the angle between the face normal $f_{normal}$ and the up vector of the coordinate system:

$$\begin{cases} f_\phi < 75°, & \text{roof surface} \\ 75° \leq f_\phi < 105°, & \text{wall surface} \\ f_\phi \geq 105°, & \text{ground surface} \end{cases}$$

Starting from a boundary surface (labeled either by semantic information transfer or by the heuristic), topological relationships are constructed by an iterative approach depicted in Figure 5-93.

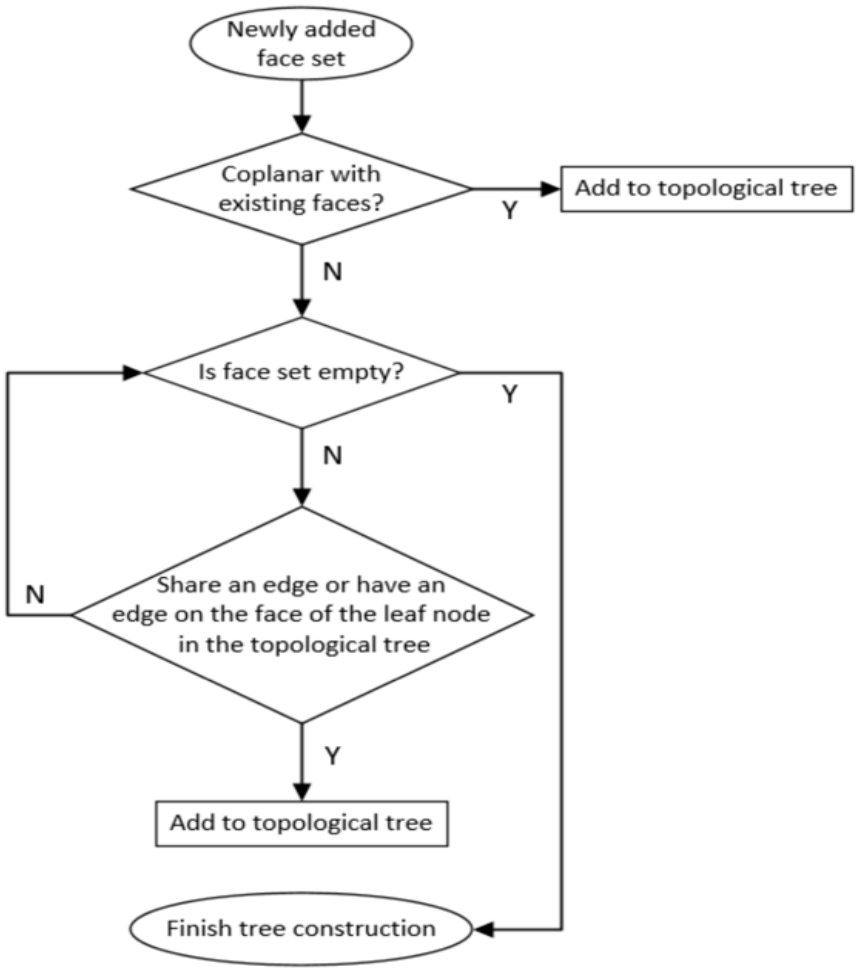



*Figure 5-93. Detect openings by topology relationships*

This iterative approach constructs a topological tree based on the edited geometry. First, all new faces which are co-planar with labeled faces are collected and added to the topological tree as the first level. Second, it is determining whether the rest of the newly added faces *share* an edge or have an edge *on the face* of the leaf node in the topological tree. If yes, we add these faces to corresponding leaf nodes. This process is repeated until there are no new faces left. Finally, all leaf nodes of the topological tree with tree depth larger than two are considered openings and classified as follows with *h* denoting the height and *w* denoting the width of the bounding box of the opening in meters. The constraints for doors are taken from [57]

$$\begin{cases} 2.0 < h < 2.5 \wedge 0.8 < w < 1.2, & \text{door} \\ \text{else}, & \text{window} \end{cases}$$

In addition, roof infrastructures generated by push-pull operations (See Figure 5-76) are not openings since in the topological tree they are children of roof surfaces.

### 5.3.4.8 Evaluation

The method is implemented as part of a standalone JavaFX based LoD3 building editor for the virtual Singapore project. The Esri procedural runtime provides facade template application and the PCL (Point Cloud Library [58] an ICP implementations as well as other utilities for point set management. Beside camera control, CityGML import/export, point cloud visualization, selection and texture assignment, the following editing operations are supported by the editor:

- Model move, scale, rotate
- Vertex, edge and face deletion
- Vertex, edge and face move
- Face and edge push-pull
- Face split
- Template application

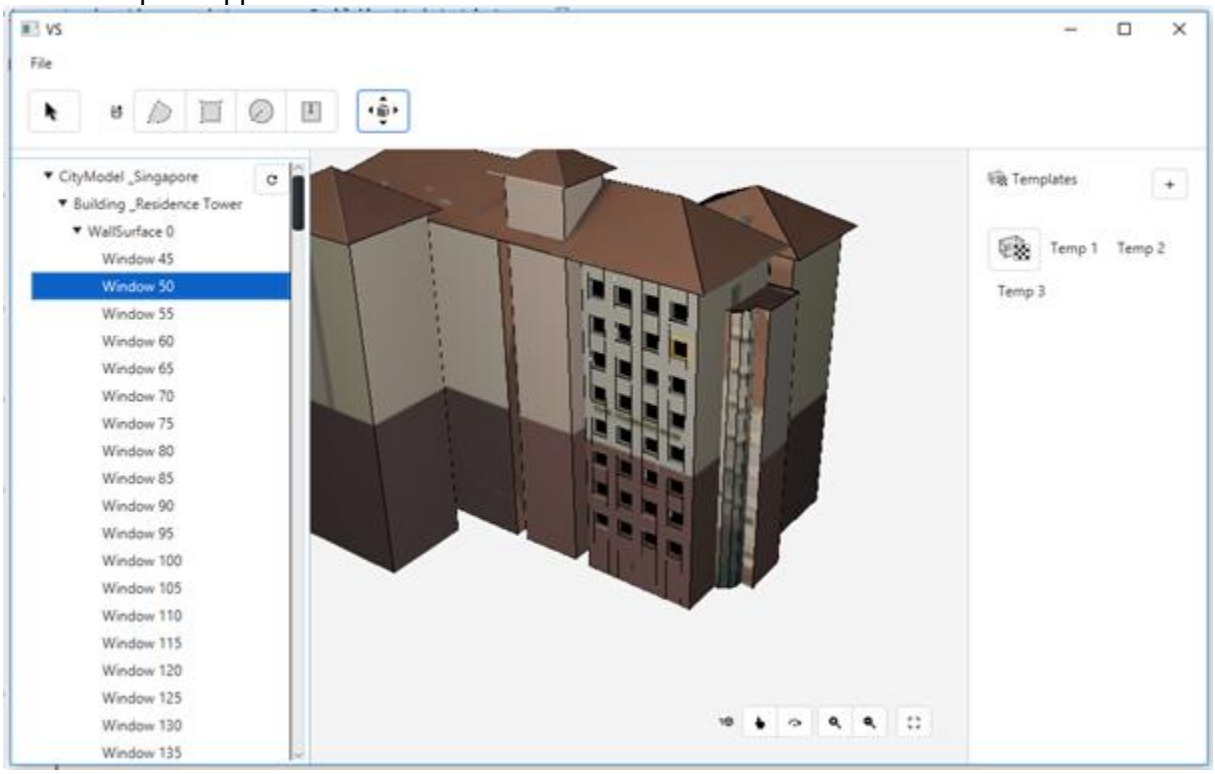



*Figure 5-94. Virtual Singapore LoD3 Editor user interface.*

We use FZKViewer as a 3rd party software to verify that the CityGML attributes are maintained when editing the building model. To illustrate typical edit operations and how the method updates the data model, two use cases are presented in the following sections: face deletion for shelters and template application for openings.

**a) Face Deletion for Shelters**

A typical example for superfluous faces and incorrect semantic information are faces which are present because of occlusions at ground level. The passageway under the shelter, shown in Figure 5-73, was constructed as a volume instead of just the roof section of the passageway. The editor allows the user to remove superfluous faces through deletion and the data model is updates accordingly (see Figure 5-95).

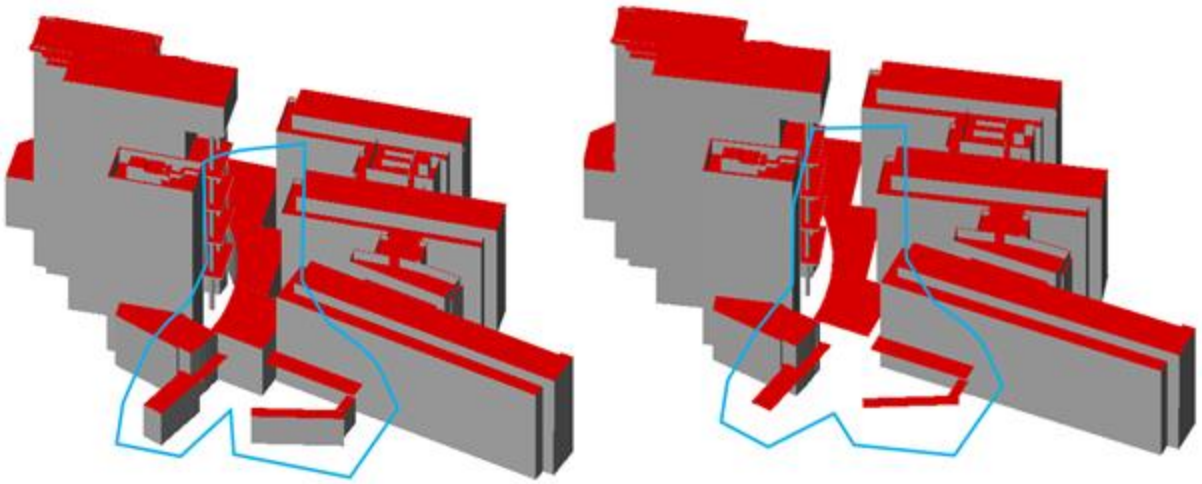


*Figure 5-95. FZKViewer: Data model update for deletion. Roofs surfaces in red, wall surfaces in grey.*

**b) Template Application for Openings**

Procedural modelling is used for adding regular structures, e.g. extending facade patterns into occluded areas or correcting measurement errors in the source model. Figure 5-96 shows hundreds of added openings for a high-rise building. Through the labeling process of the method, openings are correctly detected and classified as windows in the data model (see Figure 5-97).

*Figure 5-96. Facade template application*

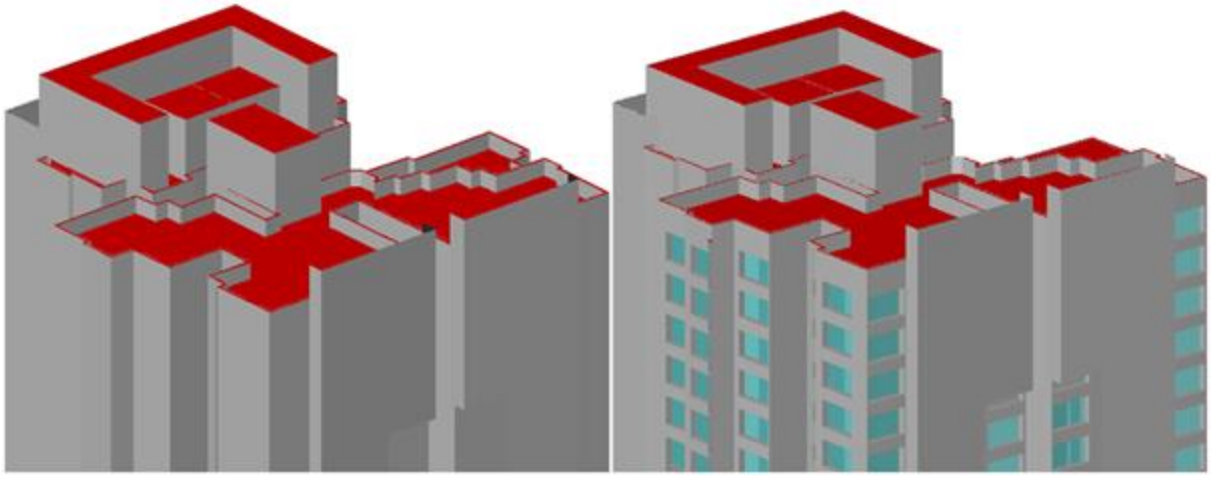


*Figure 5-97. FZKViewer: Data model update for template application. Roofs surfaces in red, wall surfaces in grey, windows in turquoise.*

These two use cases illustrate the automatic data model update process which is essential for an effective editing workflow. While the increase in efficiency, compared to manual labeling, largely depends on the editing tasks, data model consistency has shown to be one of the greatest benefits of the method.

**c) Commercial Tools**

Several commercial tools are available which support editing of CityGML models. The following sections discuss a selection of these tools in terms of their advantages and limitations regarding editing of LoD3 CityGML building models.

***CityEditor***

Trimble SketchUp is a complete and extensible 3D modelling package, widely used in architecture design and modelling. CityEditor is an extension for SketchUp, supporting import/export and editing of CityGML files. As an extension, CityEditor leverages the full capabilities of SketchUp going far beyond the editing operations that the Virtual Singapore editor supports. However, it lacks support for template applications. While loading a CityGML file, instead of storing polygons in different layers according to the semantic information, CityEditor discards all semantic information and each polygon is labelled as "unclassified". Users are able to assign semantic information for each polygon in the context menu. CityEditor also supports automatic classifications according to the angle of the polygon. The exported CityGML files contains semantic information according to users' assignment.

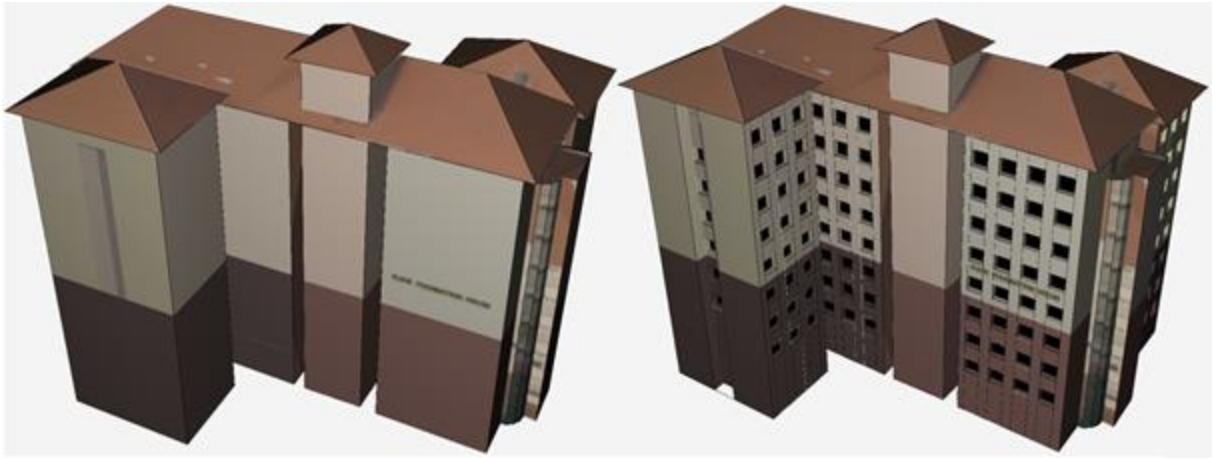


*Figure 5-98. Building Model for test cases. On the left is the original model, on the right the edited model.*

***CityGML Importer***

FME Desktop from Safe Software is a data conversion software which supports more than 450 formats and applications to help integrate and transform data. CityGML Importer is also based on Safe Software's data transformation technology and focuses on loading and converting CityGML data e.g. into .IMX files which can be edited in InfraWorks and Map 3D. While loading a CityGML file, CityGML importer maintains the semantic information of each surface by storing them in separate layers. However, the hierarchical structure of the CityGML file is lost, for instance, building parts and boundary surfaces are at different hierarchical levels in the data model but CityGML Importer regards puts them on the same level and discards the relationship. The Virtual Singapore editor maintains the semantic information as well as the hierarchical structure according to the CityGML data model.

***RhinoCity***

Rhino3D is a 3D design software which focuses on producing mathematically precise representation of curves and free-form surfaces and is widely used in architecture design and modelling.

RhinoCity [59] is an extension for Rhino3D and focuses on generation of building models and supports import/export of CityGML files. RhinoCity leverage the full editing tools from Rhino and supports many more editing operations than the Virtual Singapore editor. RhinoCity supports automatic generation of CityGML data during the 3D model creation phase but lacks the maintenance of semantic information during further editing.

***AutoCAD Map 3D***

AutoCAD is a commercial computer-aided design (CAD) and drafting software which has several domain-specific enhancements. For instance, AutoCAD Map 3D is one of AutoCAD's vertical applications. AutoCAD Map 3D incorporates geographic information system and CAD data with an industry-specific toolset for GIS and 3D mapping. AutoCAD Map 3D supports exporting drawing objects into CityGML formats. Enabling users to import/export CityGML and Google KML is one of features in 2019 release. However, CityGML data until LoD2 can be imported which means CityGML file in LoD3 cannot be imported into Map 3D. Our editor supports importing LoD3 CityGML data. Map 3D also lost semantic information in exporting to CityGML file, in the exported CityGML file, all polygons are lack of semantic labels.

*Table 5-9. Test cases for verify our model synchronization method*

| Editing Type | Editing Operation | ID |
|---|---|---|
| Transformation | Translate the entire model | 01 |
| | Rotate the entire model | 02 |
| Deletion | Delete single vertex | 03 |
| | Delete multiple vertices | 04 |
| | Delete edge between two non-planar faces | 05 |
| | Delete edge between two co-planar faces | 06 |
| | Delete face with holes | 07 |
| | Delete face without holes | 08 |
| | Delete face with child faces | 09 |
| | Delete face without child faces | 10 |
| Deformation | Move single vertices | 11 |
| | Move multiple vertices | 12 |
| | Move single edge | 13 |
| | Move multiple edges | 14 |
| | Move single face | 15 |
| | Move multiple faces | 16 |
| | Move arbitrary selection of faces / edges / vertices | 17 |
| Push-pull | Push-pull of existing face follow the face normal | 18 |
| | Push-pull on existing face with an angle | 19 |
| | Push-pull on existing face in corner vertically | 20 |
| | Push-pull on existing face in corner with an angle | 21 |
| | Push-pull on existing edge | 22 |
| | Push-pull after creation of a face on an existing face | 23 |
| | Push-pull after creation of a edge on an existing face | 24 |
| | Push-pull after creation of multiple face on an existing face | 25 |
| | Push-pull after creation of multiple face on multiple existing face | 26 |
| | Push-pull after creation of a face on an existing face with holes | 27 |
| | Push-pull after creation of a circle on an existing face | 28 |
| Template | Apply procedural templates with one level of opening extrusion | 29 |
| | Apply procedural templates with two levels of opening extrusion | 30 |

#### 5.3.4.9 Conclusions

This part introduced a data model synchronization method which preserves semantic information across editing operation. It is independent of the edit operation and depends only on geometry,

UV mappings, and materials. This enables easy integration of existing and future 3D editing tools with rich data models in a broad range of 3D applications.
The method is implemented in a LoD3 building editor for the Virtual Singapore project, including interactive push-pull and procedural generation of façades provided by 3rd party libraries. The quality of the method verified with 30 common editing tasks and compared with two commercial tools.
Currently, the focus is on interactive editing operations only. It would be interesting to see how the method performs in batch processing of 3D geometry e.g. for change detection and labeling, an important topic for maintaining and updating 3D city models.

# 6 Publications

## 6.1 International Journal Papers

[1] Xiao, Qin et al (2019) Individual Tree Detection and Crown Delineation with 3D Information from Multi-view Satellite Images. Photogrammetric Engineering & Remote Sensing [IF: 3.29]
[2] Xiao and Qin (2019) Urban land classification using oblique images. Remote Sensing [IF: 4.11] (in progress)
[3] Yao, et al, (2019) Maintaining semantic and hierarchical information across generic 3D model editing operations. Remote Sensing [IF: 4.11] (in progress)
[4] Li, et al, (2020) A Flexible Inference Machine for Wall Opening Detection and Association [IF: 4.11] (in progress)

## 6.2 Peer-reviewed Conference Papers

[1] Gruen, et al. 2019, Semantically enriched high resolution LoD3 building model generation, 3D GeoInfo Conference, Singapore.
[2] Xiao, Changlin, et al. 2019, Urban Land-Cover Classification with Façade Feature From Oblique Images, IEEE International Geoscience and Remote Sensing Symposium (IGARSS).
[3] Qin, et al. 2019, Semantic segmentation using multi-view stereo data, IEEE International Geoscience and Remote Sensing Symposium (IGARSS).
[4] Li, et al. 2018, A sliding window method for detecting corners of openings from terrestrial LiDAR data, 3D GeoInfo Conference, Delft.
[5] Qin, et al. 2019, Semantic segmentation using stereo data, IEEE International Geoscience and Remote Sensing Symposium (IGARSS)

# 7 Checklist

All implementations and supplemental materials of described approaches and methodologies can be found in our network storage \\ranau.fcl.sg\Virtual_Singapore. The checklist is shown as follows:

| **Content** | **Path** |
|---|---|
| 3 RAW DATA | DATA |
| 5.1.1 Accuracy Test of the Multi-source Input Data | WP1\code\PointProjection |

| | |
|---|---|
| 5.1.2 Semi-automatic Point Measurement | WP1\code\MeasurementTool<br>WP1\code\Geolocation |
| 5.1.3 Measurement Macros Based Reconstruction | WP1\code\MeasurementTool<br>WP1\code\FormatConverter |
| 5.1.4 3D Building Facade Segmentation | WP1\code\searchFacadePoints |
| 5.1.5 A Sliding Window Method for Detecting Facade Features | WP1\code\Cornerdetection |
| 5.1.6 Opening Detection | WP1\code\OpeningDetection |
| 5.1.7 Opening Association | |
| 5.2.1 Landcover Classification | WP2\LandcoverClassification |
| 5.2.2 Facade Semantic Segmentation | WP2\FacadeSemanticSegmentation |
| 5.2.3 Texture Mapping | WP2\TextureMapping |
| 5.3.2 Interactive post-editing | WP3\code\virtual-singapore2.0(*) |
| 5.3.3 Procedural modelling of building facades | |
| 5.3.4 Model Matching | |
| 6 PUBLICATIONS | Publications |

*WP3 developed a standalone JavaFX desktop software. The development follows the object-oriented programming in software engineering. All codes are organized in Model-View-Adapter(MVA) structure. It is suggested to find the code for Model in WP3\code\virtual-singapore2.0\src\ch\ethz\vs\data, the codes for View and Adapter in WP3\code\virtual-singapore2.0\src\ch\ethz\vs\ui.

# 8 Conclusions

WP1 aims to semi-automatically reconstruct LoD3 building models from multi-source data (e.g. 5-head images, LiDAR point clouds and LoD2 building models if they exist) in an efficient and user-friendly way. Three principles are taken into consideration when designing the workflow for semi-automated geometric building model reconstruction: 1) minimize the amount of human interaction; 2) maximize the location accuracy of vertices of the building models; 3) optimize the reliability of reconstruction. Two concepts are introduced throughout the workflow to achieve these goals: 1) develop measurement macros; 2) apply least squares matching. We have made two major corresponding achievements in this project: 1) we developed a semi-automatic point measurement method integrating a novel geometrically constrained multi-view least squares matching algorithm, which can achieve subpixel accuracy in image space and centimeter-level precision in object space for the vertices of the building models in our test dataset over the NUS campus region; 2) we developed a measurement macros based building model reconstruction tool which reconstructs the building elements in a copy-and-paste fashion, it upgrades the reconstruction style from point-based level to element-based level which significantly reduces large amount of repetition work especially for façade elements in some particular patterns. These two core developments help operators to reconstruct buildings more efficiently and more accurately, and lay a good foundation for the other work packages. Since our tested buildings, e.g. Singapore HDB buildings, residence towers, are all in regular styles, more buildings in different styles should be further involved to show the practicability and feasibility of the proposed workflow. Even though the building models reconstructed by our workflow are LOD3 models containing roof, façade and ground, no information about storey make the models unfriendly to the public and architect, further limit their application and potentiality. Thus, how

to extract the storey information from multi-source data and how to integrate the LoD3 models with other indoor information are still open issues for us.

WP2 aims to advance the automation of labeling for building attributes by leveraging multiple sources of information and has made two major achievements in this project: 1) we developed a novel feature extraction method that incorporates façade-view information from oblique images, which has improved the classification accuracy by 14.5% in our test dataset over the NUS campus region; 2) an efficient and practical workflow for labeling façade elements using deep learning approaches, which has significantly reduced the manual labeling process to a notable level. These two core developments have interacted with the other work packages in providing more reliable semantic segments for building and façade level modeling and editing. As organic components of an operable 3D LoD3 modeling workflow, well-segmented individual objects and façade elements contribute to a faster and more reliable topological modeling, and the well modeled geometry will in turn support for more accurate labeling results. Since our approach follows a machine learning paradigm, we expect that the learning models may be biased to the availability of training sets, which leads to model transferability challenges. Therefore, to enable a future seamless system for LoD3 modeling, we shall on one hand, deeply integrate the topological modeling with semantic labeling taking into consideration their organic interactions, and on the other hand develop scenario specific transfer learning method to reduce the dependency to training samples and in parallel improve the generalization capability of the semantic labeling algorithms for LoD3 building modeling.

WP3 aims to enhance the quality of LoD3 building models by an interactive 3D editor that combines user friendly editing and procedural modelling techniques. Its three major achievements are: 1) a standalone JavaFX based *LoD3 building editor* supporting camera control, CityGML import/export, point cloud visualization, model move, scale, rotate, vertex, edge and face deletion, vertex, edge and face move, face and edge push-pull and face split. 2) A *façade parametrization* tool for procedural modelling based on regular façade structures and a template library for Singapore's typical HDB façades. 3) A *model synchronization method* to maintain semantic information across editing operations, which is independent of the edit operation.
The interactive editor consumes the output from WPI1 and WP2 in CityGML format as input and allows correction and template extension e.g. in occluded areas.
The current 3D editor is a standalone application, but the façade parametrization and model synchronization methods are independent of the editor and may be integrated into 3D content creation tools such as 3DS Max, SketchUp, AutoCAD, ArcGIS Pro or CityEngine for a more streamlined workflow and a richer editing toolset. Currently, the focus is on *interactive* editing. It would be interesting to see how the methods perform in *batch processing* of 3D geometry e.g. for change detection and labelling, an important topic for maintaining and updating 3D city models.

# 9 References

[1] Harris, Christopher G., and Mike Stephens. "A combined corner and edge detector." In *Alvey Vision Conference*, vol. 15, pp. 10-5244. 1988.

[2] Hartley, Richard, and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[3] Gruen, Armin. "Adaptive least squares correlation: a powerful image matching technique," *South African Journal of Photogrammetry, Remote Sensing and Cartography,* vol. 14, pp. 175-187, 1985.

[4] Gruen, Armin, and Emmanuel P. Baltsavias. "Adaptive least squares correlation with geometrical constraints." In *Computer Vision for Robots*, vol. 595, pp. 72-82. 1986.

[5] Zhao, JiangFei, TingLei Huang, Fei Pang, and YuanJie Liu. "Genetic algorithm based on greedy strategy in the 0-1 knapsack problem." In *2009 Third International Conference on Genetic and Evolutionary Computing*, pp. 105-107. 2009.

[6] Gruen, Armin and Xinhua Wang. "News from CyberCity-modeler," *Automatic Extraction of Man-Made Objects from Aerial and Space Images (III),* pp. 93-101, 2001.

[7] Audebert, Nicolas, Bertrand Le Saux, and Sébastien Lefevre. "How useful is region-based classification of remote sensing images in a deep learning framework?," *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 5091-5094, 2016.

[8] Inglada, Jordi. "Automatic recognition of man-made objects in high resolution optical remote sensing images by SVM classification of geometric image features," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 62, pp. 236-248, 2007.

[9] Qin, Rongjun, Xin Huang, Armin Gruen, and Gerhard Schmitt. "Object-based 3-D building change detection on multitemporal stereo images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 8, pp. 2125-2137, 2015.

[10] Zhang, Qian, Rongjun Qin, Xin Huang, Yong Fang, and Liang Liu. "Classification of ultra-high resolution orthophotos combined with DSM using a dual morphological top hat profile," *Remote Sensing,* vol. 7, pp. 16422-16440, 2015.

[11] Luethje, Fritjof, Dirk Tiede, and Clemens Eisank. "Terrain extraction in built-up areas from satellite stereo-imagery-derived surface models: a stratified object-based approach," *ISPRS International Journal of Geo-Information,* vol. 6, pp. 9-21, 2017.

[12] Piltz, Björn, Steven Bayer, and Anna-Maria Poznanska. "Volume based DTM generation from very high resolution photogrammetric DSMs," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. 41, pp. 83-90, 2016.

[13] Frueh, Christian, Russell Sammon, and Avideh Zakhor. "Automated texture mapping of 3D city models with oblique aerial imagery." In *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, pp. 396-403. 2004.

[14] Douglas, David H., and Thomas K. Peucker. "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the International Journal for Geographic Information and Geovisualization,* vol. 10, pp. 112-122, 1973.

[15] Rau, Jiann-Yeou, Jyun-Ping Jhan, and Ya-Ching Hsu. "Analysis of oblique aerial images for land cover and point cloud classification in an urban environment," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 53, pp. 1304-1319, 2014.

[16] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection," *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 886–893, 2005.

[17] Crow, Franklin C. "Summed-area tables for texture mapping," in *ACM SIGGRAPH Computer Graphics*, vol. 18, pp. 207-212, 1984.

[18] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features," *IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 511-518, 2001.

[19] Huang, Xin, and Liangpei Zhang. "An adaptive mean-shift analysis approach for object extraction and classification from urban hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, pp. 4173-4185, 2008.

[20] Wu, Zhaocong, Zhongwen Hu, and Qian Fan. "Superpixel-based unsupervised change detection using multi-dimensional change vector analysis and SVM-based classification," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 7, pp. 257-262, 2012.

[21] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 234-241, 2015.

[22] Parish, Yoav IH, and Pascal Müller. "Procedural modeling of cities," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 301-308, 2001.

[23] Müller, Pascal, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. "Procedural modeling of buildings." *ACM Transactions on Graphics*, vol. 25, pp. 614-623, 2006.

[24] Lipp, Markus, Peter Wonka, and Pascal Müller. "PushPull++," *ACM Transactions on Graphics*, vol. 33, p. 130, 2014.

[25] Esri, *Procedural Runtime Whitepaper*. Available online: https://esri.github.io/esri-cityengine-sdk/html/index.html (accessed on 19 November 2019*).*

[26] Hagedorn, Benjamin, and Jürgen Döllner. "High-level web service for 3D building information visualization and analysis," in *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, pp. 1-8, 2007.

[27] Psyllidis, Achilleas, Alessandro Bozzon, Stefano Bocconi, and Christiaan Titos Bolivar. "A platform for urban analytics and semantic data integration in city planning," in *International Conference on Computer-Aided Architectural Design Futures*, pp. 21-36, 2015.

[28] Prandi, F., R. De Amicis, S. Piffer, M. Soave, S. Cadzow, E. Gonzalez Boix, and E. D'Hondt. "Using CityGML to deploy smart-city services for urban ecosystems," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, pp. 87-92, 2013.

[29] Volpi, Michele, and Vittorio Ferrari. "Structured prediction for urban scene semantic segmentation with geographic context," in *2015 Joint Urban Remote Sensing Event*, pp. 1-4, 2015.

[30] Kagal, Lalana. "A policy-based approach to governing autonomous behavior in distributed environments," *Phd. Thesis*, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 2004.

[31] Alegre, Fernando, and Frank Dellaert. "A probabilistic approach to the semantic interpretation of building facades," *International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*, 2004.

[32] Verdie, Yannick, Florent Lafarge, and Pierre Alliez. "LoD generation for urban scenes," *ACM Transactions on Graphics*, vol. 34, p. 30, 2015.

[33] Zhu, Qing, Yuan Li, Han Hu and Bo Wu. "Robust point cloud classification based on multi-level semantic relationships for urban scenes," *ISPRS journal of photogrammetry and remote sensing*, vol. 129, pp. 86-102, 2017.

[34] Wu, Chenxia, Ian Lenz, and Ashutosh Saxena. "Hierarchical Semantic Labeling for Task-Relevant RGB-D Perception.," in *Robotics: Science and Systems*, 2014.

[35] Rook, Merwin, Filip Biljecki, and A. A. Diakité. "Towards Automatic Semantic Labelling of 3D City Models," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, pp. 23-30, 2016.

[36] Sundar, Hari, Deborah Silver, Nikhil Gagvani, and Sven Dickinson. "Skeleton based shape matching and retrieval," In *Shape Modeling International*, pp. 130-139, 2003.

[37] Xie, Jin, Guoxian Dai, Fan Zhu, Edward K. Wong, and Yi Fang. "Deepshape: Deep-learned shape descriptor for 3D shape retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1335-1345, 2016.

[38] Besl, Paul J., and Neil D. McKay. "Method for registration of 3-D shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586-606. 1992.

[39] Zhang, Zhengyou. "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, pp. 119-152, 1994.

[40] Rusinkiewicz, Szymon, and Marc Levoy. "Efficient variants of the ICP algorithm.," in *IEEE Computer Society*, pp. 145-152, 2001.

[41] Fitzgibbon, Andrew W. "Robust registration of 2D and 3D point sets," *Image and Vision Computing*, vol. 21, pp. 1145-1153, 2003.

[42] Myronenko, Andriy, and Xubo Song. "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2262-2275, 2010.

[43] Ma, Jiayi, Ji Zhao, and Alan L. Yuille. "Non-rigid point set registration by preserving global and local structures," *IEEE Transactions on Image Processing*, vol. 25, pp. 53-64, 2015.

[44] Ma, Jiayi, Ji Zhao, Jinwen Tian, Alan L. Yuille, and Zhuowen Tu. "Robust point matching via vector field consensus," *IEEE Transactions on Image Processing*, vol. 23, pp. 1706-1721, 2014.

[45] AUTODESK, *AutoCAD Map 3D.* Available online: http://www.autodesk.com/autocad (accessed on 19 November 2019)

[46] TRIMBLE, *Sketchup.* Available online: www.sketchup.com (accessed on 19 November 2019)

[47] AUTODESK, *Maya.* Available online: http://www.autodesk.com/maya (accessed on 19 November 2019)

[48] Esri, *CityEngine.* Available online: https://www.esri.com/en-us/arcgis/products/esri-cityengine/overview (accessed on 19 November 2019)

[49] Esri, *ArcGIS Pro.* Available online: https://www.esri.com/en-us/arcgis/products/arcgis-pro/resources (accessed on 19 November 2019)

[50] Biljecki, Filip, Hugo Ledoux, Jantien Stoter, and Junqiao Zhao. "Formalisation of the level of detail in 3D city modelling," *Computers, Environment and Urban Systems,* vol. 48, pp. 1-15, 2014.

[51] Kolbe, Thomas H., Gerhard Gröger, and Lutz Plümer. "CityGML: Interoperable access to 3D city models," in *Geo-information for disaster management*, pp. 883-899, 2005.

[52] BentleySystems, *Bentley Map.* Available online: https://www.bentley.com/en/products/product-line/asset-performance/opencities-map (accessed on 19 November 2019)

[53] Foley, James D., Foley Dan Van, Andries Van Dam, Steven K. Feiner, John F. Hughes, Edward Angel, and J. Hughes. *Computer graphics: principles and practice*, Addison-Wesley Professional, 1996.

[54] Birdal, Tolga, and Slobodan Ilic. "A point sampling algorithm for 3d matching of irregular geometries," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6871-6878, 2017.

[55] Rodolà, Emanuele, Andrea Albarelli, Daniel Cremers, and Andrea Torsello. "A simple and effective relevance-based point sampling for 3D shapes," *Pattern Recognition Letters,* vol. 59, pp. 41-47, 2015.

[56] DeRose, Tony, Mark Meyer, and Sanjay Bakshi. "Mesh transfer using UV-space." U.S. Patent No. 8,482,569, issued July 9, 2013.

[57] Building and C. Authority, *Code of Practice on Buildable Design.* Available online: https://www.bca.gov.sg/BuildableDesign/others/copbddec00.pdf (accessed on 19 November 2019)

[58] PCL, *Point Cloud Library.* Available online: http://pointclouds.org/contact.html (accessed on 19 November 2019)

[59] RhinoTerrain, *RhinoCity,* 2019. Available online: https://www.rhinoterrain.com/en/rhinocity.html (accessed on 19 November 2019)

[60] Edelsbrunner, H. "Alpha shapes - a survey". *Tessellations in the Sciences*, pp. 1–25, 2010.

[61] Arikan, Murat, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. "O-snap: Optimization-based snapping for modeling architecture." *ACM Transactions on Graphics,* vol. 32, pp. 1-15, 2013.

[62] Dechter, Rina, and Judea Pearl. "Generalized best-first search strategies and the optimality of A." *Journal of the ACM*, pp. 505-536, 1985.

[63] Li, Jiaqiang, Biao Xiong, Filip Biljecki, and Gerhard Schrotter. "A sliding window method for detecting corners of openings from terrestrial lidar data." *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 97-103, 2018.

[64] Khoshelham, Kourosh, L. Díaz Vilariño, Michael Peter, Zhizhong Kang, and Debaditya Acharya. "The ISPRS benchmark on indoor modelling." *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* pp. 367-372, 2017.

# 10 Acknowledgements

This material is based on research/work supported by the National Research Foundation under Virtual Singapore Award No. NRF2015VSG-AA3DCM001-024. We would like to thank the Singapore Land Authority (SLA) for providing the raw data.