

Dense Point Cloud Extraction From Oblique Imagery

by

Jie Zhang

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

Nov. 20th, 2013

Signature of the Author_____

Accepted by _____
Coordinator, M.S. Degree Program Date

CHESTER F.CARLSON
CENTER FOR IMAGING SCIENCE
COLLEGE OF SCIENCE
ROCHESTER INSITITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S DEGREE THESIS

The M.S. degree Thesis of Jie Zhang
has been examined and approved by the
thesis committee as satisfactory for the
thesis requirement for the
Master of Science degree in Imaging Science

Dr. John Kerekes, Thesis Advisor

Dr. David Messinger, Committee Member

Dr. Carl Salvaggio, Committee Member

Date

Dense Point Cloud Extraction from Oblique Imagery

By

Jie Zhang

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Master of Science Degree
at the Rochester Institute of Technology

ABSTRACT

With the increasing availability of low-cost digital cameras with small or medium sized sensors, more and more airborne images are available with high resolution, which enhances the possibility in establishing three dimensional models for urban areas. The high accuracy of representation of buildings in urban areas is required for asset valuation or disaster recovery. Many automatic methods for modeling and reconstruction are applied to aerial images together with Light Detection and Ranging (LiDAR) data. If LiDAR data are not provided, manual steps must be applied, which results in semi-automated technique.

The automated extraction of 3D urban models can be aided by the automatic extraction of dense point clouds. The more dense the point clouds, the easier the modeling and the higher the accuracy. Also oblique aerial imagery provides more facade information than nadir images, such as building height and texture. So a method for automatic dense point cloud extraction from oblique images is desired.

In this thesis, a modified workflow for the automated extraction of dense point clouds from oblique images is proposed and tested. The result reveals that this modified workflow works well and a very dense point cloud can be extracted from only two oblique images with slightly higher accuracy in flat areas than the one extracted by the original workflow.

The original workflow was established by previous research at the Rochester Institute of Technology (RIT) for point cloud extraction from nadir images. For oblique images, a first modification is proposed in the feature detection part by replacing the Scale-Invariant Feature Transform (SIFT) algorithm with the Affine Scale-Invariant Feature Transform (ASIFT) algorithm. After that, in order to realize a very dense point cloud, the Semi-Global Matching (SGM) algorithm is implemented in the second modification to compute the disparity map from a stereo image pair, which can then be used to reproject pixels back to a point cloud. A noise removal step is added in the third modification. The point cloud from the modified workflow is much denser compared to the result from the original workflow.

An accuracy assessment is made in the end to evaluate the point cloud extracted from the modified workflow. From the two flat areas, subsets of points are selected from both original and modified workflow, and then planes are fitted to them, respectively. The Mean Squared Error (MSE) of the points to the fitted plane is compared. The point subsets from the modified workflow have slightly lower MSEs than the ones from the original workflow, respectively. This suggests a much more dense and more accurate point cloud can lead to clear roof borders for roof extraction and improve the possibility of 3D feature detection for 3D point cloud registration.

TABLE OF CONTENTS

ABSTRACT.....	ii
LIST OF FIGURES.....	vi
LIST OF TABLES.....	xi
1 Introduction.....	1
1.1 Three-Dimensional (3D) Modeling and Reconstruction	1
1.1.1 Creation of Building Models	2
1.1.2 Texturing of the Building Models.....	3
1.2 Point Cloud Extraction.....	4
1.3 Organization of Thesis.....	6
2 Objective.....	7
3 Data	9
3.1 Pictometry Imagery.....	9
3.2 Some Testing Images.....	12
3.3 Pictometry Imagery vs. Testing Images	13
4 Methodology	14
4.1 Previous Work	14
4.1.1 The RIT 3D Workflow.....	14
4.1.2 Fundamental Algorithms	16
4.2 First Modification – Affine Scale-Invariant Feature Transform (ASIFT).....	34
4.2.1 The Workflow after the First Modification	35
4.2.2 Affine Scale-Invariant Feature Transform (ASIFT) Algorithm	35

4.3 Second Modification - Semi-Global Matching (SGM)	40
4.3.1 The Workflow after the Second Modification	40
4.3.2 Image Rectification	41
4.3.3 Semi-Global Matching (SGM) Algorithm	43
4.3.4 Dense Point Cloud Projection from Disparity Map	53
4.4 Third Modification - Noise Removal	55
4.4.1 The Workflow after the Third Modification	56
4.4.2 Noise Removal Methods	56
5 Results	62
5.1 Results of the Previous Work	62
5.2 Results of the First Modification – ASIFT	63
5.2.1 Results of Original Size Images	64
5.2.2 Results of Focal Length Fixation	67
5.3 Results of the Second Modification – SGM	71
5.3.1 Results of Image Rectification	72
5.3.2 Results of Calculation of Disparity Maps	73
5.3.3 Results of Dense Point Cloud from Disparity Maps	76
5.4 Results of the Third Modification - Noise Removal	83
5.5 Accuracy Assessment	87
6 Conclusion	95
7 Future Work	97
References	99

LIST OF FIGURES

Figure 3-1: The images from five perspectives provided by Pictometry.....	9
Figure 3-2: All images used in this thesis. (a) ten North-viewing images, (b) six West-viewing images, (c) nine nadir viewing images.....	10
Figure 3-3: Another pair of Pictometry images.....	11
Figure 3-4: The Tsukuba images.....	12
Figure 3-5: The toys images.....	12
Figure 3-6: The City Hall images.....	13
Figure 3-7: The block images. (a) is the Mega block image, (b) is the Lego block image.....	13
Figure 4-1: The RIT workflow.....	15
Figure 4-2: The calculation of Gaussian images and DOG images.....	18
Figure 4-3: The detection of maximum and minimum in the DOG image.....	18
Figure 4-4: The computation of the histograms of the gradient magnitude and orientation at each image sample point in a region around the keypoint location.....	20
Figure 4-5: The workflow of RANSAC.....	22
Figure 4-6: The epipolar geometry.....	26
Figure 4-7: The pseudo code of the complete LM algorithm.....	29
Figure 4-8: The pseudo code of algorithm for solving the sparse normal equations.....	33
Figure 4-9: Initial modified workflow with ASIFT to detect the features.....	34
Figure 4-10: Large deformation of the same building in the images taken from cameras with a slight orientation change.....	36
Figure 4-11: The projective camera model.....	37

Figure 4-12: Geometric interpretation of the decomposition.....	38
Figure 4-13: Overview of the AISFT algorithm.....	38
Figure 4-14: The sampling of the parameters $\theta = \arccos \frac{1}{t}$ and ϕ	39
Figure 4-15: The workflow after the second modification using SGM algorithm for dense stereo matching.....	40
Figure 4-16: The comparison of different coordinate systems in Bundler and OpenCV.....	41
Figure 4-17: The aggregation of cost through 16 paths. (a) is the minimum cost path $L_r(p, d)$, (b) is the 16 paths from all directions r for pixel p	47
Figure 4-18: The workflow of SGM algorithm.....	49
Figure 4-19: Merging all tiles by calculating a weighted mean at overlapping areas.....	52
Figure 4-20: The model used for calculating the 3D coordinates in basic photogrammetry.....	53
Figure 4-21: The workflow after the third modification with noise removal.....	56
Figure 4-22: The diagram of the statistical removal method.....	57
Figure 4-23: The diagram of the radius outlier removal method.....	58
Figure 5-1: (a) is the small region around Imaging Science Building in the size of 1000×1000 pixels. (b) is the point cloud of 10 North-viewing small region images.....	62
Figure 5-2: The comparison of ASIFT and SIFT to find the correspondences in oblique images. (a) is the image pair. (b) is the correspondences computed by ASIFT and SIFT: left of (b) is the result of ASIFT, right of (b) is the result of SIFT.....	63
Figure 5-3: The point cloud of ten small regions of North-viewing Pictometry images. (a) is the result of ASIFT workflow (44640 vertices), (b) is the result of SIFT workflow (34358 vertices).....	64

Figure 5-4: The errors on the walls in both point clouds. (a) is the point cloud from ASIFT workflow, (b) is the point cloud form SIFT workflow.....	65
Figure 5-5: The point clouds of original size images. (a) is the result from ASIFT workflow (506,084 vertices), (b) is the result from SIFT workflow (357,702 vertices).....	66
Figure 5-6: The vertical walls after going back to the original size images.....	66
Figure 5-7: The floating walls marked by red points.....	67
Figure 5-8: The diagram of the input and output focal lengths of Bundler, comparing to the actual focal length.....	68
Figure 5-9: Wall errors decrease when the value of parameter constrain_focal_weight increases. (a), (b), (c) and (d) are the point clouds of constrain_focal_weight with the value of 1.0e-4, 1.0e6, 1.0e12 and 1.0e24, respectively.....	70
Figure 5-10: Sparse point cloud extracted by ASIFT workflow.....	72
Figure 5-11: Image rectification result. (a) and (b) is the left and right images for rectification. (c) is the rectification result.....	73
Figure 5-12: The diagram of disparity shift of features with different distance to the camera....	74
Figure 5-13: The disparity maps of Tsukuba images by both SGM and SGBM approaches. (a) is the image, (b) is the ground truth, (c) is the result from SGBM, (d) is the result from SGM.....	74
Figure 5-14: The disparity maps of toys images by SGBM function embedded in OpenCV. (a) is the images after rectification, (b) is the disparity map.....	75
Figure 5-15: The disparity map of Pictometry oblique images computed by SGBM in OpenC...	75
Figure 5-16: The dense point clouds of toys images from different view angles.....	76

Figure 5-17: The comparison of the point clouds extracted by the original workflow and the workflow of the second modification. (a) is the point cloud from the second modification. (b) is the point cloud from the original workflow.....	77
Figure 5-18: The dense point cloud of Pictometry images from different view angles. (a) is the point cloud before bilateral filtering. (b) is the point cloud after bilateral filtering.....	78
Figure 5-19: The dense point cloud extracted from toys images with different disparity ranges. (a) is extracted from image pair with lower disparity range, (b) is extracted from image pair with higher disparity range.....	79
Figure 5-20: The dense point clouds. (a) is from Mega blocks, (b) is from Lego blocks.....	80
Figure 5-21: The disparity maps of Tsukuba images by both SGM and SGBM approaches. (a) is the image, (b) is the ground truth, (c) is the result from SGBM, (d) is the result from SGM.....	81
Figure 5-22: The disparity maps of toys images by both SGM and SGBM approaches. (a) is the rectified images, (b) is the result from SGBM, and (c) is the result from SGM.....	82
Figure 5-23: The disparity maps of Pictometry images by both SGM and SGBM approaches. (a) is the images after rectification, (b) is the results from SGBM, and (c) is the results from SGM.....	83
Figure 5-24: The point cloud projected from the disparity map computed by SGM. (a) is the result by SGBM, (b) is the result by SGM.....	83
Figure 5-25: Noise in the point cloud. (a) shows the extraneous points floating besides the walls and roofs, (b) shows invalid points exist through the ray of light.....	84
Figure 5-26: The results of noise removal. (a) is the point cloud with noise, (b) is the result of statistical removal method, (c) is the result of radius outlier removal method.....	85

Figure 5-27: The comparison of the details of the two noise removal methods. (a) is the detail of the statistical removal method, (b) is the detail of the radius outlier removal method.....	85
Figure 5-28: The noise removal results in sparse areas. (a) and (b) are the results of statistical removal method and radius outlier removal, respectively, in West-viewing point cloud, (c) and (d) are the results in vertical point cloud.....	86
Figure 5-29: Dense point cloud after noise removal by statistical removal method. (a) is the point cloud with noise, (b) is the point cloud after noise removing by statistical removal method.....	87
Figure 5-30: Figure 5-30 The point subsets from the point clouds extracted by the original workflow and the modified workflow. (a) are the corresponding image parts, (b) are the point subsets from the point clouds extracted by the original workflow, (c) are the point subsets from the point clouds extracted by the modified workflow.....	89
Figure 5-31 The inliers and outliers for each subset points after fitting by RANSAC with the same parameters. (1a) is the RANSAC results for the up plane (original workflow). (1b) is the error histogram in Euclidian distance for the up plane (original workflow). (2a) is the RANSAC results for the down plane (original workflow). (2b) is the error histogram in Euclidian distance for the down plane (original workflow). (3a) is the RANSAC results for the up plane (modified workflow). (3b) is the error histogram in Euclidian distance for the up plane (modified workflow). (4a) is the RANSAC results for the down plane (modified workflow). (4b) is the error histogram in Euclidian distance for the down plane (modified workflow).....	90

LIST OF TABLES

Table 5-1: Focal lengths for the ten North-viewing images after SBA.....	68
Table 5-2: Different output focal lengths with different values for constrain_focal_weight.....	69
Table 5-3: The vertices decrease when the value of constrain_focal_weight goes up.....	69
Table 5-4: The comparison of the original SGM and SGBM in OpenCV.....	81

1 Introduction

With the increasing availability of low-cost digital cameras with small or medium sized sensors, airborne images of urban regions have been used in various applications, such as building detection (Sirmacek and Unsalan, 2008), building modeling (Jurisch and Mountain, 2008; Wang et al., 2008; Smith et al., 2009; Habbecke and Kobbelt, 2010), surface reconstruction (Wu et al., 2012), road extraction (Amo et al., 2006; Zhang et al., 2011), shadow compensation (Tsai, 2006), and vegetation extraction (Secord and Zakhor, 2007). In all of these, building modeling is a most common technique. Correct and consistent representations of buildings are required for asset valuation or disaster recovery. Currently, aerial images combined with Light Detection and Ranging (LiDAR) data are used to realize fully automated techniques for the extraction of building geometry (Wang et al., 2008; Haala and Kada, 2010; Cheng et al., 2011). However, image-based modeling still remains the most complete, economical, portable, flexible and widely used approach (Remondino and El-Hakim, 2006) for urban mapping. So, a robust and automated technique based only on images is desired.

1.1 Three-Dimensional (3D) Modeling and Reconstruction

3D modeling and reconstruction of an object is a process that starts from data acquisition and ends with a 3D virtual model visually interactive on a computer, and is a long-lasting research problem in the graphic, vision and photogrammetric communities (Remondino and El-Hakim, 2006). In the photogrammetric field, many methods have been proposed to create 3D buildings

from airborne images recently. Most of them mentioned in the literature reconstruct the building in two steps: create the building models and add texture to the building.

1.1.1 Creation of Building Models

The creation of building models is the first step and the most difficult step in 3D modeling. Mostly, it is solved in a manual or semi-automatic way. For example, Jurisch and Mountain (2008) create the geometric model manually. They first captured the 2D building polygons from ortho images, and LiDAR data were used for orthorectification and tessellation. And then they manually measured the building heights based on oblique images by using a height measurement tool. Finally, they extruded the 2D building polygons into a 3D block model. Smith et al. (2009) extract the geometry semi-automatically. They first manually measure the roof structures and then automatically extrude to the ground which was defined by a manual point measurement.

In order to find the footprint automatically, Habbecke and Kobbelt (2010) registered the oblique aerial images with cadastral maps which contain the footprints of buildings. They said that oblique images provide information on building heights, appearance of facades, and terrain elevation, but challenges are introduced by the scale of pixels varying across an image caused by perspective foreshortening, the strongly changing appearance between different views, and the inevitable (self-) occlusion of buildings. After registration, a valid height map was generated from the oblique images with the camera parameters computed during the registration. Based on this height map, they built models on the footprints on the registered cadastral map.

Wang et al. (2008) extracted buildings automatically. However, they derived building models from LiDAR data instead of aerial images saying that the occlusions and shadows that occur in the images may fail the extraction. Because of point spacing, scanning angle, the performance of the line extraction algorithm, they refined the derived building models by projecting back on the vertical image and triangulating with accurate ground control points. An affine transformation was used to correct the building models with the parameters estimated by using the distance between the projected roof edges and the extracted edges from the image.

1.1.2 Texturing of the Building Models

After establishing the building models, the next step is texturing. Jurisch and Mountai (2008) achieve it in a manual way. They first extract the most suitable image for each face from the image data set, and then crop the appropriate section from the rectified image. Finally, they apply the cropped image to each face. Smith et al. (2009) perform the texturing of the 3D geometry from oblique images automatically. They used the in-flight GPS and rotation information to calibrate the cameras in the coordinate system of the geometric model, and then calculated the corresponding image coordinates for each vertex of a triangle mesh representing the 3D surface by knowing the parameters of the interior and exterior orientation of the cameras. At last they attached color values within the projected triangle to the surface.

Wang et al. (2008) first selected the best oblique image before texturing. Since their oblique images are captured at a certain angle, they defined a reference vector with a certain angle to the building façade within the vertical plane passing through the normal vector of the facade. Then

they assign a score to all oblique images based on the angle between the reference vector and the vector from the center of the façade to the camera center of an oblique image. The image with highest score was chosen for texturing. At the same time, a visibility analysis is performed to make sure that the façade is not blocked by other buildings. After that, based on exterior orientation parameters from the GPS/IMU, they computed the accurate exterior orientation parameters from the differences between the building façade projected onto the oblique image and the building edges on the image. At last, the right image portion was determined by projecting the building façade onto the oblique image with the corrected exterior orientation parameters and added to the 3D building model.

1.2 Point Cloud Extraction

Extracting a dense point cloud of the structures, based on some 2D images taken from different view angles, is another common approach for building models creation. The key to automatically recovering 3D structure from aerial images is to identify reliable invariant features, match these features from images with diverse angular views of that scene, and then generate accurate mathematical relationships to relate the images (Walli et al., 2009).

Walli et al. (2009) and Nilosek and Salvaggio (2010) implemented computer vision techniques to reconstruct a scene from airborne nadir-viewing images. They first establish the corresponding relationships of the semi-invariant features detected between images by the Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) algorithm, and then remove erroneous matches by using the RANdom SAmple Consensus (RANSAC) (Fischler and Bolles, 1981) technique in

conjunction with the Fundamental Matrix relationship between images of the same scene. Once the correct corresponding points have been found, they use the Sparse Bundle Adjustment (SBA) (Lourakis and Argyros, 2004) algorithm to compute and optimize the camera parameters and 3D coordinates. At the end, they recovered a dense point cloud from matching images by using the Fundamental Matrix to reduce the correspondence search and a normalized cross correlation to detect the correspondence. The approach by Agarwal et al. (2009), more focusing on the processing speed, is different from the previous one. However, the fundamental procedure for structure recovery is the same as previously mentioned: SIFT, RANSAC, Fundamental Matrix and SBA.

The images used in the work done by Nilosek and Salvaggio (2010) and Walli et al. (2009) for finding the corresponding points are nadir images, which have less perspective transformation between images taken from the same scene than oblique images. Also, they have much more overlapping percentage. The larger the overlap, the easier to find the corresponding points. But oblique images have a much higher degree of affine and projective transform than nadir iamges, which limits the accuracy of the SIFT algorithm in detecting the corresponding features.

Gerke (2009) did not use the vertical images, focusing instead on the potential of the oblique views only. In their preprocessing steps, they first calibrated the cameras and then rectified images by the camera positions and orientations. After that, they compute the disparity map of stereo image pair by using the Semi-Global-Matching (SGM) approach (Hirschmuller, 2008). From the disparity map, a very dense 3D point cloud was derived. There are some manual steps in the camera calibration stage. They manually derived measured tie points, and also ground

control points (GCP) were added into the bundle block adjustment (Gerke and Nyaruhuma, 2009).

The goal of this present study is trying to find a fully automated method for dense point cloud extraction from oblique images only.

1.3 Organization of Thesis

This thesis is organized as follows: the objective is outlined in Section 2, while the experimental data are described in Section 3. In Section 4, the basic method and fundamental algorithms are detailed, and also some modifications are described for this study. Results are discussed in Section 5. Conclusion follows in Section 6, and the future work is in Section 7.

2 Objective

As discussed in Section 1, almost all existing methods for 3D modeling and reconstruction are based on nadir images or are semi-automated when using oblique images. In contrast to nadir images, oblique aerial images, taken at an oblique angle with respect to the ground, have the important advantage of providing information on building facades, such as height and texture. This information enables new kinds of applications such as 3D city modeling and damage assessment (Gerke and Kerle, 2011). However, the oblique images have significantly varying image scale and more occlusion from buildings or high trees, which create much more difficulties in processing.

The objective of this project is to take advantage of oblique images and extract dense point clouds in an automated way. The extracted dense point cloud, instead of the LiDAR data, can be used to create the building models. Furthermore, adding texture to the building models can realize an automated method for building modeling only using aerial images. Also, the dense point cloud gives higher possibility for 3D point cloud registration. It gathers all facades information for a building, even if some walls are not complete or do not exist in a particular point cloud. A roof frame, or surface, can be extracted from dense point cloud for asset evaluation or disaster recovery.

In this thesis, the approach will be to make several modifications to an existing workflow. In order to take the advantage of oblique images, a new algorithm will be implemented to detect the

features from oblique images. It is an affine invariant mechanism which can detect features in images with large differences in view angles, such as oblique images we used in this project. Then, in order to extract dense point cloud, an efficient stereo matching will be used to compute the disparity map, from which a dense point cloud can be derived. At last, a noise removal step will be added to remove noise from the extracted dense point cloud.

3 Data

In recent years, oblique aerial images have become widely available, such as “bird’s-eye view” in Microsoft’s internet map service, “Maps and Earth” in Google, and Pictometry Online (Gill, 2010). The oblique airborne images used in this study were Pictometry images, provided by Pictometry, Inc.

3.1 *Pictometry Imagery*

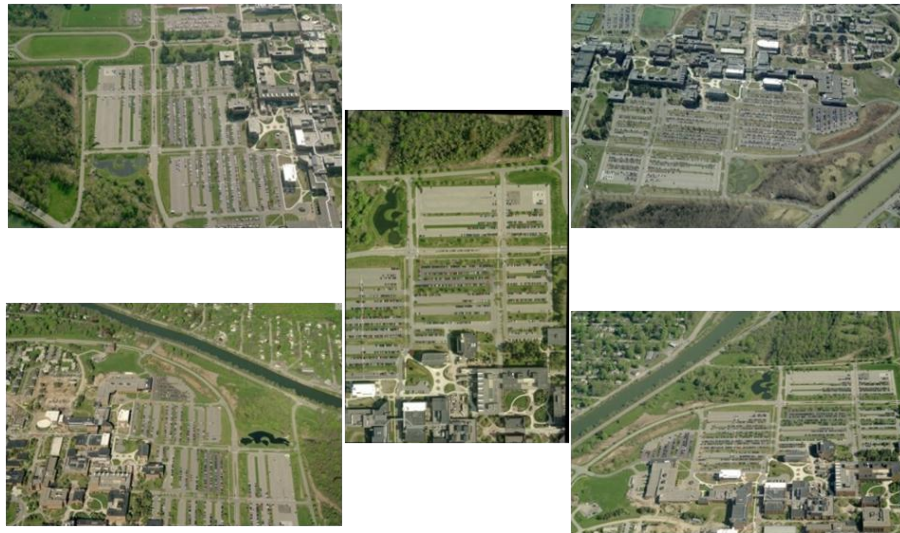


Figure 3 - 1 The images from five perspectives provided by Pictometry.

Pictometry data include five perspectives (as Figure 3-1 shows) and the system provides position and orientation data, suggesting ready referencing and photogrammetric processing (Gerke and Kerle, 2011). The ground resolution for the oblique imagery is approximately 14-18cm with the flying height between approximately 1378m and 1420m. The ground resolution for the nadir

imagery is approximately 14cm. The pixel size at the focal plane is 0.0074mm with a nominal focal length for the vertical camera of 65mm and the oblique cameras of 85mm. The overlap for



(a)



(b)



(c)

Figure 3 - 2 All images used in this thesis. (a) ten North-viewing images, (b) six West-viewing images, (c) nine nadir viewing images.

the vertical images varies approximately from 30% to 60% and the overlap of the oblique imagery is approximately from 20% to 90% (Smith et al., 2009).

Figure 3-2 gives all the images used in this thesis: ten North-viewing images, six West-viewing images, and nine nadir viewing images. The data site is the Rochester Institute of Technology (RIT) campus, which includes many buildings and parking lots with cars on them. These images, with the size of 3248×4872 pixels, are taken at a height around 1400m with focal lengths between 11400 and 11500 pixels.

The vertical image is taken by positioning the camera view vertically to the earth. It shows almost all the roof information of the buildings but no façade information at all. The other oblique images are taken from oblique view directions: North, South, West or East, which show not only the roof information but also the façades of the buildings.

Here is another pair of Pictometry images from another site (Figure 3-3). They are taken at a height of almost 800m with the focal length of almost 22904 pixels.



Figure 3 - 3 Another pair of Pictometry images.

3.2 Some Testing Images

Some additional testing images are used in this project, such as the Tsukuba images (Figure 3-4), the toys images (Figure 3-5), the City Hall images (Figure 3-6) and the block images (Figure 3-7). The Tsukuba images, in the size of 288×384 pixels, are provided by Scharstein and Szeliski (2002). The toys images and the block images, in the size of 2448×3264 pixels, are taken by the author with an iPhone 4S with a focal length of 3070 pixels. The City Hall images, in the size of 5616×3744 pixels, are provided by Hover Inc. (2013).



Figure 3 - 4 The Tsukuba images.



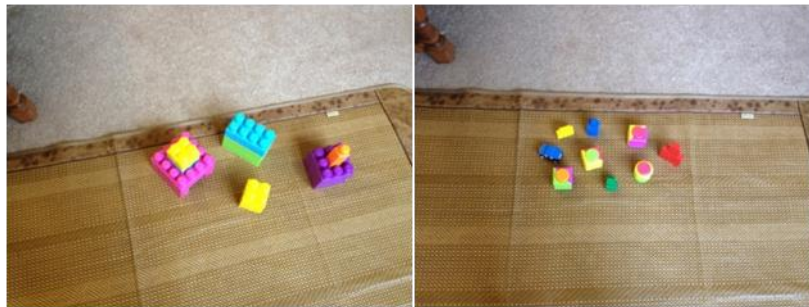
Figure 3 - 5 The toys images.



Figure 3 - 6 The City Hall images.

3.3 Pictometry Imagery vs. Testing Images

Compared to the Pictometry images, the Tsukuba images are rectified. They have high overlap percentage and low disparity range. The toys images are oblique images with high ratio of toy height to camera height. The City Hall images are oblique airline images with high ratio of building height to camera height. The block images are oblique images with different ratio of block height to camera height. These differences will be seen to affect the accuracy of the results.



(a)

(b)

Figure 3 - 7 The block images. (a) is the Mega block image, (b) is the Lego block image.

4 Methodology

In this thesis, several modifications will be made to an existing workflow. First, the previous work by RIT researchers will be introduced. Then, the first modification will be made by replacing the feature detection part from the Scale-Invariant Feature Transform (SIFT) algorithm to the Affine Scale-Invariant Feature Transform (ASIFT) algorithm in order to detect the features more accurately and efficiently in oblique images. After that, the second modification will be made by implementing Semi-Global Matching (SGM) algorithm to do the stereo matching. At last, the third modification is adding a noise removal method to remove the extraneous points in the extracted point cloud.

4.1 Previous Work

4.1.1 The RIT 3D Workflow

RIT researchers Nilosek and Salvaggio (2010) proposed a workflow to generate 3D point cloud based on some common computer vision techniques. This workflow is constructed in four parts: feature detection and camera pose estimation, sparse 3D reconstruction and optimization, geo-rectification, and dense model extraction. After that, Professor Harvey Rhody established a similar workflow for processing nadir imagery of downtown Rochester, NY, taken from RIT's Wildfire Airborne Sensor Program (WASP) sensor (WASP, 2013). This similar workflow is shown as Figure 4-1.

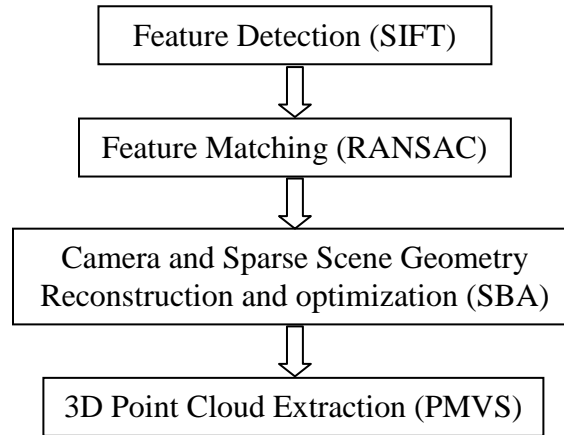


Figure 4-1 The RIT workflow.

In this workflow, the Scale-Invariant Feature Transform (SIFT) algorithm is used to detect the distinctive keypoints in each image. These feature points are invariant to scale, rotation, translation, and slight changes in illumination. Then, the RANdom SAMple Consensus (RANSAC) algorithm is applied to match the keypoints between all the images and remove outliers.

After that, they implement Bundler (Snavely et al., 2006) to compute and optimize the camera parameters and 3D coordinates. Bundler is a structure-from-motion (SfM) system. It takes a set of images, image features, and image matches to produce 3D reconstructions of camera and sparse scene geometry, using a modified version of the Sparse Bundle Adjustment (SBA) as the underlying optimization engine.

At last, Post-Match Vacancy Service (PMVS) (Furukawa and Ponce, 2009) is implemented to reproject image pixels back to the 3D world. PMVS is a multi-view stereo software. It takes a set

of images and camera parameters to reconstruct 3D structure of an object or a scene visible in the images, presenting the results as a set of oriented points containing both 3D coordinate and the surface normal.

4.1.2 Fundamental Algorithms

In the RIT workflow, there are some fundamental algorithms, such as SIFT, RANSAC, Fundamental matrix and SBA.

4.1.2.1 Scale-Invariant Feature Transform (SIFT)

Image matching is the first step in 3D reconstruction from stereo images. SIFT (Lowe, 2004), the Scale Invariant Feature Transform, was proposed by David Lowe in 1999 (Lowe, 1999). It can robustly identify distinctive invariant features from images. These features are invariant to image scale, rotation, and partially to illumination viewpoint. They can be used to perform reliable matching between different views of an object or scene. Furthermore we can use these corresponding features to calculate the camera parameters.

The SIFT algorithm computes the features in the following four major stages: scale-space extrema detection, keypoint localization, orientation assignment, and keypoint descriptor.

A. Detection of Scale-Space Extrema

SIFT utilizes a Difference of Gaussian (DOG) edge detector of varying widths to identify candidate locations and simulate all the possible scales. One of the reasons is because it is an efficient way to compute, and most importantly, DOG provides a close approximation to the

scale-normalized Laplacian of Gaussian, $\sigma^2 \nabla^2 G$, which is required for true scale invariance, mentioned by Lindeberg (1994).

The Difference-of-Gaussian (DOG) function convolved with the image of two nearby scales separated by a constant multiplicative factor k is

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma), \quad (4-1)$$

where the scale space, $L(x, y, \sigma)$, of an image, $I(x, y)$, is

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \quad (4-2)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}. \quad (4-3)$$

Figure 4-2 shows the efficient approach to construction of $D(x, y, \sigma)$. In the left column, the image is repeatedly convolved with Gaussians with varying width to produce Gaussian images. Lowe (2004) divided each octave of scale space, s , into, k , intervals, such that $k = 2^{1/s}$. For each octave, the Gaussian image count should be $s + 3$ to guarantee that the extrema detection covers a complete octave. In the right column, the DOG images are produced by subtraction of adjacent Gaussian images. After finishing one octave, the calculation is repeated by down-sampling the Gaussian image by a factor of 2.

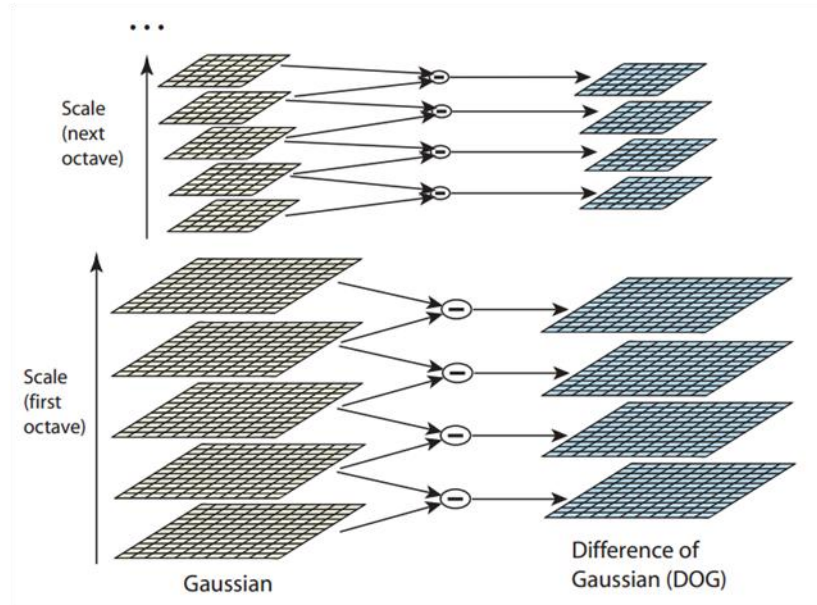


Figure 4 - 2 The calculation of Gaussian images and DOG images (Lowe, 2004).

In the DOG images, each sample point is compared to its twenty-six neighbors, eight neighbors in the current images and nine neighbors in above and below images, to detect the local maxima and minima, shown as Figure 4-3.

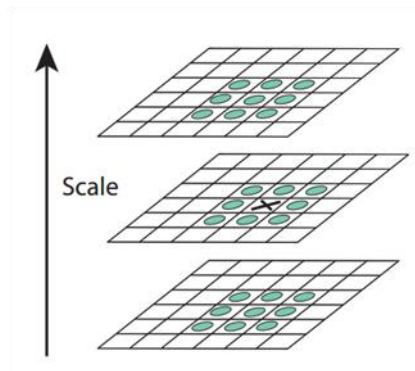


Figure 4 - 3 The detection of maximum and minimum in the DOG image (Lowe, 2004).

The frequency of sampling in the space and scale domains is determined by studying a range of sampling frequencies. As a result, Lowe (2004) chooses to use 3 scale samples per octave, and $\sigma = 1.6$ in the image domain.

B. Accurate Keypoint Localization

All the keypoint candidates are located at the central sample point. However, the matching accuracy and stability would be highly improved if the location of the maximum was interpolated by 3D quadratic fitting, proposed by Brwon and Lowe (2002). Shift the origin to the sample point, then the scale-space fuction, $D(x, y, \sigma)$ is expanded by Taylor expansion up to the quadratic terms as

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial^2 \mathbf{x}^2} \mathbf{x}, \quad (4-4)$$

where D and its derivatives are evaluated at the sample point and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point. Taking the derivative of $D(\mathbf{x})$ with respect to \mathbf{x} gives the location of the extrema, $\hat{\mathbf{x}}$

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2} \frac{\partial D}{\partial \mathbf{x}}. \quad (4-5)$$

Add this final offset, $\hat{\mathbf{x}}$, to the location of its sample point to get the location of the extrema.

Because of the strong response along edges by the DOG function, an additional threshold on the ratio of principal curvatures is set to eliminate the edge keypoints.

C. Orientation assignment

In order to achieve rotation invariance, a consistent orientation of each keypoint is calculated based on local image properties. At the selected scale of the location of keypoint, compute the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, for each image sample $L(x, y)$ in a region of the keypoint, as Figure 4-4 shows,

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}, \quad (4-6)$$

$$\theta(x, y) = \tan^{-1}((L(x, y + 1) - L(x, y - 1)) / (L(x + 1, y) - L(x - 1, y))). \quad (4-7)$$

Then, an orientation histogram was established for each keypoint. The orientation histogram is weighted by $m(x, y)$ and a Gaussian-weighted circular window. After that, a dominant direction is selected by searching the peak in the orientation histogram. This direction is the direction of this keypoint.

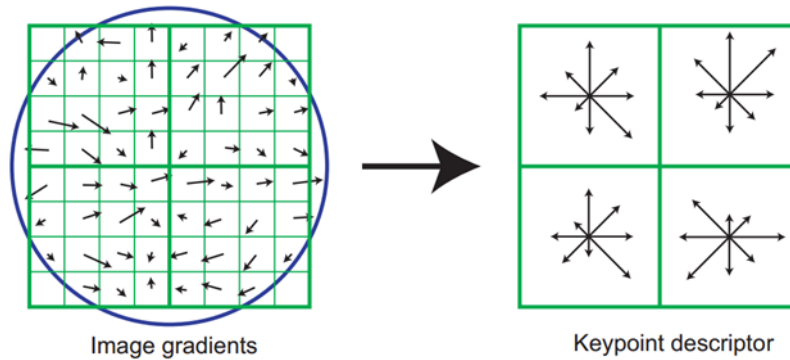


Figure 4 - 4 The computation of the histograms of the gradient magnitude and orientation at each image sample point in a region around the keypoint location (Lowe, 2004).

D. The local image descriptor

Now we have the location, scale, and orientation for each keypoint. The next step is to form a descriptor for the keypoint which is highly distinctive and as invariant as possible to illumination and 3D viewpoint. Lowe (2004) proposed that a 4×4 array of histograms with 8 orientation bins in each achieved the best results, not the 2×2 array as shown in Figure 4-4. Therefore, each keypoint descriptor contains $4 \times 4 \times 8 = 128$ feature elements.

In order to achieve illumination invariance, the vector is normalized to unit length to reduce the effects of contrast change. For non-linear illumination changes, first threshold the unit feature values no larger than 0.2 and then renormalize.

4.1.2.2 RANdom Sample Consensus (RANSAC)

RANSAC has proven to be a robust technique for outlier removal, even in the presence of large numbers of incorrect matches (Hartley and Zisserman, 2004). This paradigm is particularly applicable to the feature matching problem because local features detected by SIFT would often make mistakes.

RANSAC proposed by Fischler and Bolles (1981) is a paradigm for fitting a model to experimental data, rather than an interpretation of sensed data in terms of a set of predefined models. The later optimizes the fit of a model to all of the presented data based on smoothing assumption in its parameter estimation problem, which has no internal mechanisms for detecting

and rejecting gross errors. Or it uses as much of the data as possible to obtain an initial solution and then single out one gross error in heuristics if the smoothing assumption does not hold.

However, RANSAC is very different from the conventional smoothing techniques, capable of smoothing data that contain a significant percentage of gross errors. The workflow of RANSAC is (shown as Figure 4-5):

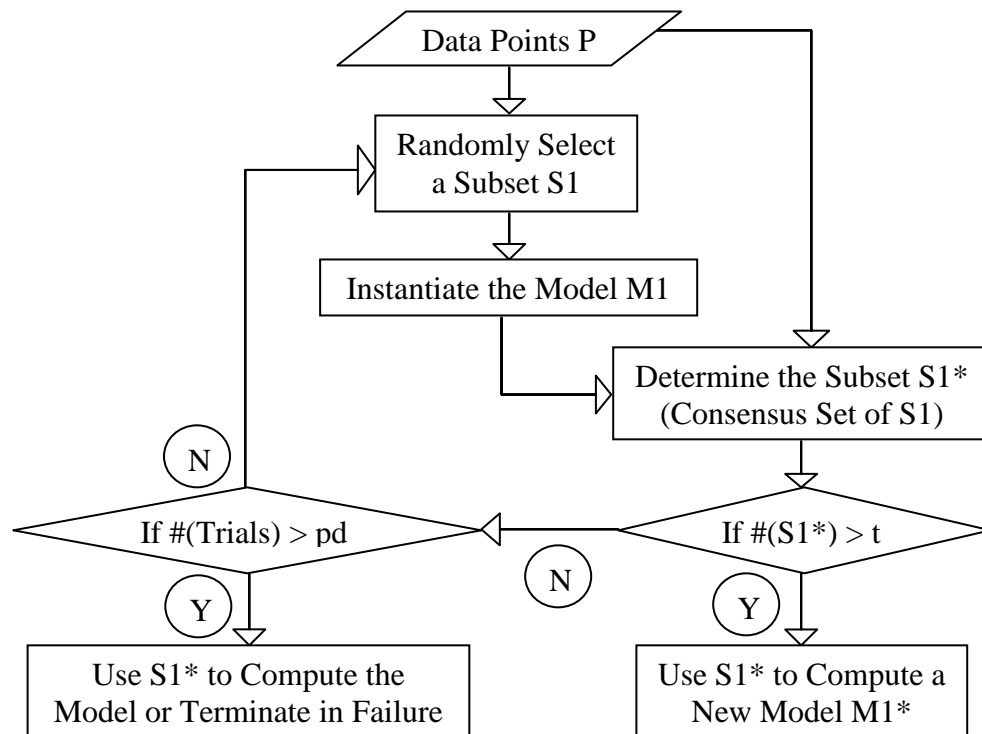


Figure 4 - 5 The workflow of RANSAC.

- Given a set of data points P , randomly elect a subset $S1$ of n data points. This n is the minimum data points required by the selected model to instantiate its free parameters.
- Instantiate the model $M1$ using the subset $S1$.

- c) Determine the subset $S1^*$, called the consensus set of $S1$, in P which are within some error tolerance of the instantiated model $M1$.
- d) Check the number of points in the subset $S1^*$. If it is greater than some threshold t , then use $S1^*$ to compute a new model $M1^*$.
- e) Or check the number of trials. If the trial number is smaller than a threshold pd , go back to a) to select a new subset $S2$ randomly.
- f) Or solve the model with the largest consensus set found, or terminate in failure.

The model in this project is taken from the computer vision community (Nilosek and Salvaggio, 2010). For the two images looking at the same object from two different views, a point in one image will correspond to a line in the other image, in the following relationship:

$$Fx_1 = l_2. \quad (4-8)$$

Here F is the 3×3 fundamental matrix, x_1 is a homogeneous point in image 1, and l_2 is the epipolar line in image 2. And the corresponding point in image 2 lies on the epipolar line as

$$x_2^T l_2 = 0. \quad (4-9)$$

So the relationship of the two correspondence points with the fundamental matrix would be

$$x_2^T F x_1 = 0. \quad (4-10)$$

Any two matching features from SIFT must obey this equation.

4.1.2.3 Fundamental Matrix

In order to obtain 3D information from images taken from different views, there are only two approaches. The first one is to compute the 3×4 projection matrix which relates pixel coordinates to 3D coordinates. However, it needs to know the internal and external geometry of both the two cameras and the rigid displacement between them, which is not always possible. The other approach, using projective information, only requires the relationship between the different viewpoints. This relationship is called the Fundamental matrix (Luong and Faugeras, 1996).

A. The Projective Model

Considering a pinhole camera, the model performs a perspective projection of an object point M onto a pixel m in the retinal plane through the optical center C . The optical axis goes through C and is perpendicular to the retinal plane at point c . In the orthonormal system of the retinal plane, called normalized coordinates, the center at c is (c, u, v) in another 3D orthonormal system of coordinates centered at the optical center C . The two axes of the 3D coordinate are parallel to the retinal ones and the third one is parallel to the optical axis (C, x, y, z) . The relationship between the coordinates, m and M , in these two systems of coordinates is

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ T \end{bmatrix}. \quad (4-11)$$

Here U , V and S are the homogeneous coordinates of the pixel m and X , Y , Z , T are the homogeneous coordinates of the point M . In a matrix form:

$$\mathbf{m} = \tilde{\mathbf{P}}\mathbf{M}, \quad (4-12)$$

where $\tilde{\mathbf{P}}$ is the 3×4 projection matrix. So the relationship between the world coordinates and the pixel coordinates is linear projective, which is independent of the choice of the coordinate systems.

Since the homogeneous representation of camera center C satisfies the equation

$$\tilde{\mathbf{P}}\tilde{\mathbf{C}} = \mathbf{0}. \quad (4-13)$$

If we decompose $\tilde{\mathbf{P}}$ as $[\mathbf{P}\mathbf{p}]$, and decompose $\tilde{\mathbf{C}}$ as $[\mathbf{C}^T \ 1]^T$, the camera center C is

$$\mathbf{C} = -\mathbf{P}^{-1}\mathbf{p}. \quad (4-14)$$

B. The Epipolar Geometry and the Fundamental Matrix

Consider two images taken by two cameras looking at the same scene, as Figure 4-6 shows. They are both linear projections.

In Figure 4-6, C and C' are the optical centers of the first and second cameras, respectively. Project the line $\langle C, C' \rangle$ to the first image R in a point e , and to the second image R' in a point e' . These two points e and e' are the epipoles. All the lines in the first image through e and in the second image through e' are epipolar lines. In stereovision, for a point m in the first retina, its

corresponding point m' in the second retina would lie on its epipolar line l'_m , and vice versa.

The relationship between the point m and its projection l'_m is projective linear.

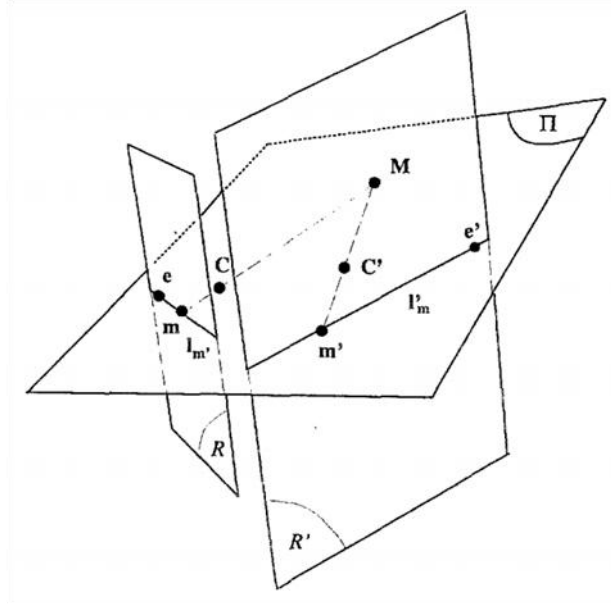


Figure 4 - 6 The epipolar geometry (Luong and Faugeras, 1996).

In the case of uncalibrated cameras, define a 3×3 Fundamental matrix \mathbf{F} to describe relationship of m and l'_m , we have

$$l'_m = \mathbf{F}m. \quad (4-15)$$

And the corresponding point m' lies on the line l'_m , then

$$m'^T \mathbf{F}m = 0. \quad (4-16)$$

By reversing the role of the two images, we have

$$\mathbf{m}^T \mathbf{F}^T \mathbf{m}' = 0. \quad (4-17)$$

The epipole is the project the optical center C of the first camera into the second camera:

$$\mathbf{e}' = \tilde{\mathbf{P}}' \begin{bmatrix} -\mathbf{P}^{-1} \mathbf{p} \\ 1 \end{bmatrix} = \mathbf{p}' - \mathbf{P}' \mathbf{P}^{-1} \mathbf{p}. \quad (4-18)$$

And the point of infinity of $\langle C, M \rangle$ projected by the second camera is

$$\tilde{\mathbf{P}}' \begin{bmatrix} \mathbf{P}^{-1} \mathbf{m} \\ 0 \end{bmatrix} = \mathbf{P}' \mathbf{P}^{-1} \mathbf{m}. \quad (4-19)$$

So the epipolar line of m of the first retina is obtained by taking the cross-product of epipole and the point of infinity of $\langle C, M \rangle$:

$$\mathbf{l}'_m = [\mathbf{p}' - \mathbf{P}' \mathbf{P}^{-1} \mathbf{p}] \times \mathbf{P}' \mathbf{P}^{-1} \mathbf{m} = [\mathbf{p}' - \mathbf{P}' \mathbf{P}^{-1} \mathbf{p}]_{\times} \mathbf{P}' \mathbf{P}^{-1} \mathbf{m}. \quad (4-20)$$

Hence, the Fundamental matrix represented by the perspective projection, $\tilde{\mathbf{P}}$, in the two-cameras case is

$$\mathbf{F} = [\mathbf{p}' - \mathbf{P}' \mathbf{P}^{-1} \mathbf{p}]_{\times} \mathbf{P}' \mathbf{P}^{-1}. \quad (4-21)$$

4.1.2.4 Sparse Bundle Adjustment (SBA)

In the RIT 3D workflow, SBA (Lourakis and Argyros, 2004) is an essential step to compute and optimize the camera parameters and 3D coordinates in the scene, because of a large number of unknowns contributing to the minimized reprojection error. It is an advanced version of Bundle Adjustment (BA) (Triggs et al., 2000) with low computational costs.

A. Bundle Adjustment (BA)

BA has been commonly used in the field of photogrammetry in last decade. It is a technique to obtain a reconstruction by refining the 3D structure and the intrinsic and extrinsic parameters of cameras simultaneously.

BA minimizes the reprojection error between the observed and predicted image points by using a non-linear least squares algorithm named Levenberg-Marquardt (LM). LM linearizes the function to be minimized in the neighborhood of the current estimate iteratively, which is computationally very demanding when there are many parameters. Fortunately, the matrix in the linear systems involved has a sparse block structure. Therefore, a lower computational cost strategy can be used by taking advantage of the zeroes pattern.

B. The Levenberg-Marquardt (LM) Algorithm

The LM algorithm is a standard technique for non-linear least-squares problems. It iteratively minimizes the sum of squares of non-linear real-valued functions with multi variants. LM behaves like a combination of steepest descent and the Gauss-Newton method. The pseudo code of complete LM algorithm is in Figure 4-7. For details, the interested reader is referred to (Lourakis and Argyros, 2004) for more comprehensive treatments.

C. Sparse Bundle Adjustment (SBA)

In order to deal with the problem of bundle adjustment efficiently, the LM algorithm is developed to a large extent based on the presentation regarding SBA.

Input: A vector function $f: \mathcal{R}^m \rightarrow \mathcal{R}^n$ with $n \geq m$, a measurement vector $\mathbf{x} \in \mathcal{R}^n$ and an initial parameters estimate $\mathbf{p}_0 \in \mathcal{R}^m$.

Output: A vector $\mathbf{p}^+ \in \mathcal{R}^m$ minimizing $\|\mathbf{x} - f(\mathbf{p})\|^2$.

Algorithm:

```

k := 0; v := 2;  $\mathbf{p} := \mathbf{p}_0$ ;
 $\mathbf{A} := \mathbf{J}^T \mathbf{J}$ ;  $\epsilon_p := \mathbf{x} - f(\mathbf{p})$ ;  $\mathbf{g} := \mathbf{J}^T \epsilon_p$ ;
stop := ( $\|\mathbf{g}\|_\infty \leq \epsilon_1$ );  $\mu := \tau * \max_{i=1, \dots, m} (A_{ii})$ ;
while (not stop) and ( $k < k_{\max}$ )
    k := k+1;
    repeat
        Solve  $(\mathbf{A} + \mu \mathbf{I}) \delta_p = \mathbf{g}$ ;
        if ( $\|\delta_p\| \leq \epsilon_2 \|\mathbf{p}\|$ )
            stop := true;
        else
             $\mathbf{p}_{\text{new}} := \mathbf{p} + \delta_p$ ;
             $\rho := (\|\epsilon_p\|^2 - \|\mathbf{x} - f(\mathbf{p}_{\text{new}})\|^2) / (\delta_p^T (\mu \delta_p + \mathbf{g}))$ ;
            if  $\rho > 0$ 
                 $\mathbf{p} = \mathbf{p}_{\text{new}}$ ;
                 $\mathbf{A} := \mathbf{J}^T \mathbf{J}$ ;  $\epsilon_p := \mathbf{x} - f(\mathbf{p})$ ;  $\mathbf{g} := \mathbf{J}^T \epsilon_p$ ;
                Stop := ( $\|\mathbf{g}\|_\infty \leq \epsilon_1$ );
                 $\mu := \mu * \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$ ;  $v := 2$ ;
            else
                 $\mu := \mu * v$ ;  $v := 2 * v$ ;
            endif
        endif
    until ( $\rho > 0$ ) or (stop)
endwhile

```

Figure 4 - 7 The pseudo code of the complete LM algorithm (Lourakis and Argyros, 2004).

Assume that n 3D points are seen in m images with different viewpoints. \mathbf{x}_{ij} is the projection of the i -th point on image j . BA was implemented to find the set of parameters, including intrinsic and extrinsic matrices for cameras, to most accurately predict the locations of the observed n

points from m available images. If \mathbf{a}_j represents the parameters of camera j , and \mathbf{b}_i represents the 3D point i , the reprojection error would be minimized by BA as

$$\min_{\mathbf{a}_j, \mathbf{b}_i} \sum_{i=1}^n \sum_{j=1}^m d(\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i), \mathbf{x}_{ij})^2, \quad (4-22)$$

where $\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)$ denotes the predicted projection of point i on image j , and $d(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between the inhomogeneous image points represented by \mathbf{x} and \mathbf{y} . If κ and λ are the dimensions of each \mathbf{a}_j and \mathbf{b}_i , respectively, the total number of minimization parameters is $m\kappa + n\lambda$.

Let $\mathbf{P} = (\mathbf{a}_1^T, \dots, \mathbf{a}_m^T, \mathbf{b}_1^T, \dots, \mathbf{b}_n^T)^T$, $\mathbf{P} \in \mathcal{R}^M$ describes all parameters of m projection matrices and n 3D points, $\mathbf{X} = (\mathbf{x}_{11}^T, \dots, \mathbf{x}_{1m}^T, \mathbf{x}_{21}^T, \dots, \mathbf{x}_{2m}^T, \dots, \mathbf{x}_{n1}^T, \dots, \mathbf{x}_{nm}^T)^T$, $\mathbf{X} \in \mathcal{R}^N$ represents the measured image point coordinates across all cameras, and $\hat{\mathbf{X}}$ generated from a function $\hat{\mathbf{X}} = f(\mathbf{P})$ as $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_{11}^T, \dots, \hat{\mathbf{x}}_{1m}^T, \hat{\mathbf{x}}_{21}^T, \dots, \hat{\mathbf{x}}_{2m}^T, \dots, \hat{\mathbf{x}}_{n1}^T, \dots, \hat{\mathbf{x}}_{nm}^T)^T$ defines estimated measure with $\hat{\mathbf{x}}_{ij} = \mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)$. Therefore, BA is minimizing the squared Mahalanobis distance $\epsilon^T \Sigma_X^{-1} \epsilon$ with $\epsilon = \mathbf{X} - \hat{\mathbf{X}}$ over \mathbf{P} , which could be solved by using LM algorithm to iteratively solve the weighted normal equations

$$\mathbf{J}^T \Sigma_X^{-1} \mathbf{J} \delta = \mathbf{J}^T \Sigma_X^{-1} \epsilon, \quad (4-23)$$

where \mathbf{J} is the Jacobian of f and δ is the desired update to the parameter vector \mathbf{P} . The normal equations in Equation (4-23) have a regular sparse block structure which results from the lack of interaction between parameters of different cameras and different 3D points.

Suppose there are $n=4$ points visible in $m=3$ images, then the measured vector of image point coordinates is $\mathbf{X} = (\mathbf{x}_{11}^T, \mathbf{x}_{12}^T, \mathbf{x}_{13}^T, \mathbf{x}_{21}^T, \mathbf{x}_{22}^T, \mathbf{x}_{23}^T, \mathbf{x}_{31}^T, \mathbf{x}_{32}^T, \mathbf{x}_{33}^T, \mathbf{x}_{41}^T, \mathbf{x}_{42}^T, \mathbf{x}_{43}^T)^T$, and the parameter vector is $\mathbf{P} = (\mathbf{a}_1^T, \mathbf{a}_2^T, \mathbf{a}_3^T, \mathbf{b}_1^T, \mathbf{b}_2^T, \mathbf{b}_3^T, \mathbf{b}_4^T)^T$. Let \mathbf{A}_{ij} and \mathbf{B}_{ij} denote $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ and $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$, respectively, the Jacobian \mathbf{J} is

$$\frac{\partial \mathbf{X}}{\partial \mathbf{P}} = \begin{pmatrix} A_{11} & 0 & 0 & B_{11} & 0 & 0 & 0 \\ 0 & A_{12} & 0 & B_{12} & 0 & 0 & 0 \\ 0 & 0 & A_{13} & B_{13} & 0 & 0 & 0 \\ A_{21} & 0 & 0 & 0 & B_{21} & 0 & 0 \\ 0 & A_{22} & 0 & 0 & B_{22} & 0 & 0 \\ 0 & 0 & A_{23} & 0 & B_{23} & 0 & 0 \\ A_{31} & 0 & 0 & 0 & 0 & B_{31} & 0 \\ 0 & A_{32} & 0 & 0 & 0 & B_{32} & 0 \\ 0 & 0 & A_{33} & 0 & 0 & B_{33} & 0 \\ A_{41} & 0 & 0 & 0 & 0 & 0 & B_{41} \\ 0 & A_{42} & 0 & 0 & 0 & 0 & B_{42} \\ 0 & 0 & A_{43} & 0 & 0 & 0 & B_{43} \end{pmatrix}. \quad (4-24)$$

And let the covariance matrix is

$$\Sigma_{\mathbf{X}} = \text{diag}(\Sigma_{x11}, \Sigma_{x12}, \Sigma_{x13}, \Sigma_{x21}, \Sigma_{x22}, \Sigma_{x23}, \Sigma_{x31}, \Sigma_{x32}, \Sigma_{x33}, \Sigma_{x41}, \Sigma_{x42}, \Sigma_{x43}). \quad (4-25)$$

The Equation (4-23) becomes

$$\begin{pmatrix} U_1 & 0 & 0 & W_{11} & W_{21} & W_{31} & W_{41} \\ 0 & U_2 & 0 & W_{12} & W_{22} & W_{32} & W_{42} \\ 0 & 0 & U_3 & W_{13} & W_{23} & W_{33} & W_{43} \\ W_{11}^T & W_{12}^T & W_{13}^T & V_1 & 0 & 0 & 0 \\ W_{21}^T & W_{22}^T & W_{23}^T & 0 & V_2 & 0 & 0 \\ W_{31}^T & W_{32}^T & W_{33}^T & 0 & 0 & V_3 & 0 \\ W_{41}^T & W_{42}^T & W_{43}^T & 0 & 0 & 0 & V_4 \end{pmatrix} \begin{pmatrix} \delta_{a1} \\ \delta_{a2} \\ \delta_{a3} \\ \delta_{b1} \\ \delta_{b2} \\ \delta_{b3} \\ \delta_{b4} \end{pmatrix} = \begin{pmatrix} \epsilon_{a1} \\ \epsilon_{a2} \\ \epsilon_{a3} \\ \epsilon_{b1} \\ \epsilon_{b2} \\ \epsilon_{b3} \\ \epsilon_{b4} \end{pmatrix}, \quad (4-26)$$

with $\mathbf{U}_j = \sum_{i=1}^4 \mathbf{A}_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \mathbf{A}_{ij}$, $V_i = \sum_{j=1}^3 \mathbf{B}_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \mathbf{B}_{ij}$, $\mathbf{W}_{ij} = \mathbf{A}_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \mathbf{B}_{ij}$, $\epsilon_{a_j} = \sum_{i=1}^4 \mathbf{A}_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij}$,

and $\epsilon_{b_i} = \sum_{j=1}^3 \mathbf{B}_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij}$.

Equation (4-26) can be expressed as

$$\begin{pmatrix} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \epsilon_a \\ \epsilon_b \end{pmatrix}, \quad (6-27)$$

and multiply it by the block matrix

$$\begin{pmatrix} \mathbf{I} & -\mathbf{WV}^{*-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \quad (4-28)$$

The result is

$$\begin{pmatrix} \mathbf{U}^* - \mathbf{WV}^{*-1} \mathbf{W}^T & \mathbf{0} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \epsilon_a - \mathbf{WV}^{*-1} \epsilon_b \\ \epsilon_b \end{pmatrix}. \quad (4-29)$$

Noting that the top right block of the left hand matrix is zero, δ_a can be determined by

$$(\mathbf{U}^* - \mathbf{WV}^{*-1} \mathbf{W}^T) \delta_a = \epsilon_a - \mathbf{WV}^{*-1} \epsilon_b. \quad (4-30)$$

Input: The current parameter vector partitioned into m camera parameter vectors \mathbf{a}_j and n 3D point parameter vectors \mathbf{b}_i , a function \mathbf{Q} employing the \mathbf{a}_j and \mathbf{b}_i to compute the predicted projections $\hat{\mathbf{x}}_{ij}$ of the i -th point on the j -th image, the observed image point locations \mathbf{x}_{ij} and a damping term μ for LM.

Output: The solution δ to the normal equations involved in LM-based bundle adjustment.

Algorithm:

Compute the derivative matrices $A_{ij} := \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j} = \frac{\partial Q(\mathbf{a}_j, \mathbf{b}_i)}{\partial \mathbf{a}_j}$, $B_{ij} := \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i} = \frac{\partial Q(\mathbf{a}_j, \mathbf{b}_i)}{\partial \mathbf{b}_i}$

and the error vectors $\epsilon_{ij} := \mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}$,

where i and j assume values in $\{1, \dots, n\}$ and $\{1, \dots, m\}$ respectively.

Compute the following auxiliary variables:

$$\begin{aligned} U_j &:= \sum_i A_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} A_{ij} & V_i &:= \sum_j B_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} B_{ij} & W_{ij} &:= A_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} B_{ij} \\ \epsilon_{a_j} &:= \sum_i A_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij} & \epsilon_{b_i} &:= \sum_j B_{ij}^T \Sigma_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij} \end{aligned}$$

Augment U_j and V_i by adding μ to their diagonals to yield U_j^* and V_i^* .

Compute $Y_{ij} := W_{ij} V_i^{*-1}$.

Compute δ_a from $\mathbf{S}(\delta_{a_1}^T, \delta_{a_2}^T, \dots, \delta_{a_m}^T)^T = (\mathbf{e}_1^T, \mathbf{e}_2^T, \dots, \mathbf{e}_m^T)^T$, where \mathbf{S} is a matrix consisting of $m \times m$ blocks; block jk is defined by $S_{jk} = \delta_{jk} U_j^* - \sum_i Y_{ij} W_{ik}^T$, where δ_{jk} is Kronecker's delta and

$$\mathbf{e}_j = \epsilon_{a_j} - \sum_i Y_{ij} \epsilon_{b_i}.$$

Compute each δ_{b_i} from the equation $\delta_{b_i} = V_i^{*-1} (\epsilon_{b_i} - \sum_j W_{ij}^T \delta_{a_j})$.

Form δ as $(\delta_a^T, \delta_b^T)^T$.

Figure 4 - 8 The pseudo code of algorithm for solving the sparse normal equations (Lourakis and Argyros, 2004).

Afterword, ϵ_b can be computed by

$$\mathbf{V}^* \delta_b = \epsilon_b - \mathbf{W}^T \delta_a. \quad (4-31)$$

All of the above solution could be directly generalized to arbitrary n and m . The general procedure is summarized in Figure 4-8, which can be embedded into the LM algorithm at the point indicated by the rectangular box in Figure 4-7, realizing a complete SBA algorithm.

4.2 First Modification – Affine Scale-Invariant Feature Transform (ASIFT)

In the RIT 3D workflow, SIFT is used to detect feature descriptors from the aerial nadir images. However, this project is trying to extract point cloud from oblique images, with much more affine transformations in the images from the different views. Therefore a better algorithm should be used to find feature points on oblique images, which are distinguished and could be matched easily with higher accuracy. Morel and Yu (2009) proposed an affine invariant algorithm which an extension of the SIFT method.

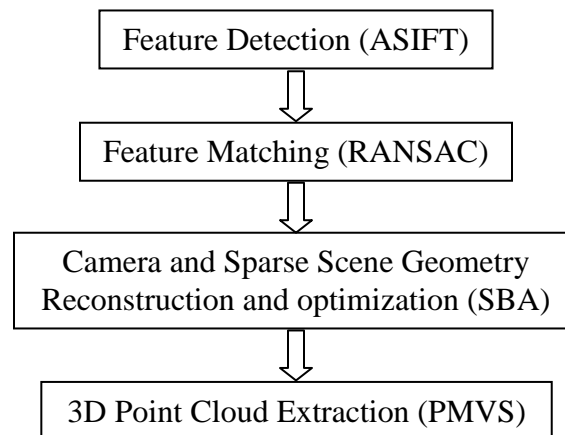


Figure 4 - 9 Initial modified workflow with ASIFT to detect the features.

4.2.1 The Workflow after the First Modification

After the first modification, the workflow is as Figure 4-9 shows. In the new workflow, ASIFT, instead of SIFT, is used to detect the keypoints. The feature points detected by ASIFT are not only invariant to scale, rotation, translation, and illumination but also invariant to affine transformation.

The other three steps are the same as the RIT 3D workflow. It applies RANSAC to match the keypoints between all the images and remove outliers, and it implements Bundler to compute and optimize the camera parameters and 3D coordinates. At last, PMVS is used to reproject image pixels back to the 3D world.

4.2.2 Affine Scale-Invariant Feature Transform (ASIFT) Algorithm

In stereo matching, oblique images taken by different cameras from different viewpoints contain significant deformation. Figure 4-10 shows the large deformation possible with a slight change of the camera orientation, although both two images are taken from cameras viewing the same direction.

ASIFT estimates not only scale but also camera axis orientation parameters, latitude and longitude angles, and then normalizes rotation and translation. It is an affine invariant extension of SIFT, by covering all the possible orientations for the camera, and then use SIFT to detect the keypoints.

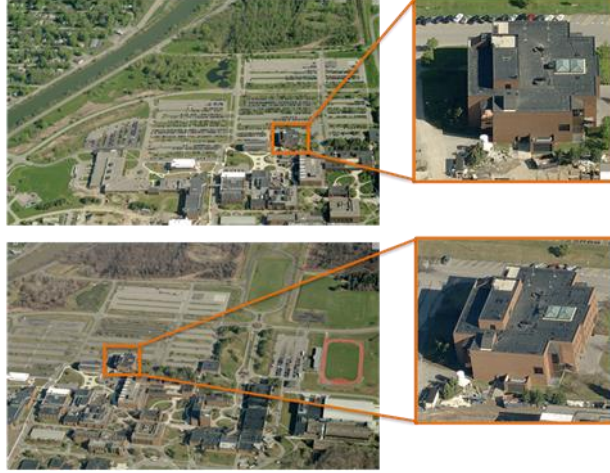


Figure 4 - 10 Large deformation of the same building in the images taken from cameras with a slight orientation change

4.2.2.1 Affine Camera Model and Tilts

As illustrated by Figure 4-11, the digital image acquisition of a flat object is

$$\mathbf{u} = \mathbf{S}_1 \mathbf{G}_1 \mathbf{A} \mathbf{T} \mathbf{u}_0, \quad (4-32)$$

where \mathbf{u} is a digital image, \mathbf{u}_0 is an infinite resolution frontal view of the flat object, \mathbf{T} is a plane translation, \mathbf{A} is a planar projective map, \mathbf{G}_1 is Gaussian convolution modeling the optical blur, and \mathbf{S}_1 is the standard sampling operator. More generally, the apparent deformation of a solid object caused by a change of camera position can be locally approximated by affine transforms.

Morel and Yu (2009) mentioned a theorem that any affine map $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ with a strictly positive determinant which is not a similarity has a unique decomposition

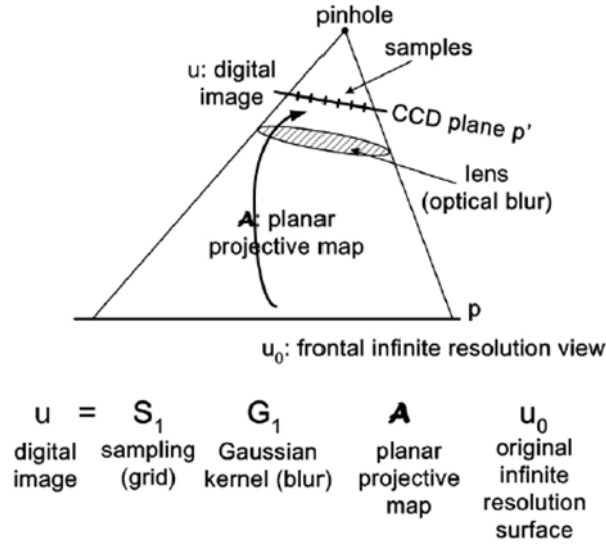


Figure 4 - 11 The projective camera model (Morel and Yu, 2009)

$$A = H_\lambda R_1(\psi) T_t R_2(\phi) = \lambda \begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix}, \quad (4-33)$$

where zoom parameter $\lambda > 0$, λt is the determinant of A , R_i are rotations, $\phi \in [0, \pi)$, and T_t is a tilt, namely a diagonal matrix with first eigenvalue $t > 1$ and the second one equal to 1.

Figure 4-12 is showing the decomposition of the affine map via the theorem. Assume the camera is far away from the scene and starts from a frontal view. In the observation hemisphere, the angle ϕ is called longitude, and $\theta = \arccos(1/t)$ is latitude. The camera can rotate with the angle ψ around its optical axis. Also, it can move forward or backward as described by zoom parameter λ .

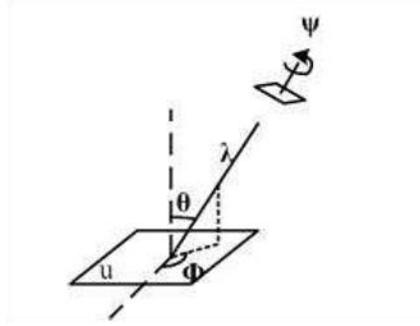


Figure 4 - 12 Geometric interpretation of the decomposition (Morel and Yu, 2009).

4.2.2.2 Algorithm steps

ASIFT first estimates all distortions caused by possible variation of the camera orientations and then uses SIFT to finish the keypoints detection, outlined in Figure 4-13. That means it estimates three parameters: the scale, the camera longitude angle and the latitude angle and normalizes the other three: the two translation parameters and the rotation angle.

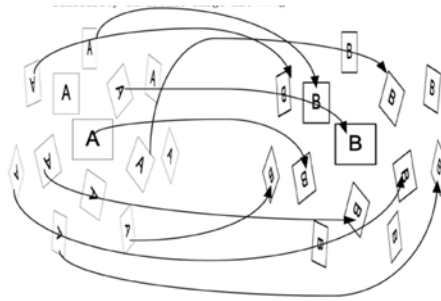


Figure 4 - 13 Overview of the ASIFT algorithm (Morel and Yu, 2009).

ASIFT proceeds by the following steps:

1. Considering all the possible longitude ϕ and latitude θ of the camera orientation from a frontal position, transform each image to estimate the affine distortions. The images are rotated by an angle ϕ and tilted with parameter $t = \left| \frac{1}{\cos \theta} \right|$. The tilt is a directional t-sampling in digital images, convoluting by a Gaussian with standard deviation $c\sqrt{t^2 - 1}$, where $c = 0.8$.
2. A finite and small number of latitude and longitude angles are considered. However, these angles are well sampled to ensure to keep as close as possible to all possible views. Sample the latitude angle θ so that the associated tilts follow a geometric series $1, a, a^2, \dots, a^n$, with $a = \sqrt{2}$, and n goes up to 5 or more. Consequently, transition tilts between different images goes up to 32. Sample the longitude angle ϕ for each tilt follow an arithmetic series $0, \frac{b}{t}, \dots, \frac{kb}{t}$, with $b \approx 72^\circ$ and k is the lat integer when $\frac{kb}{t} < 180$. On the observation hemisphere, the sampling of the parameters $\theta = \arccos \frac{1}{t}$ and ϕ is illustrated by Figure 4-14.

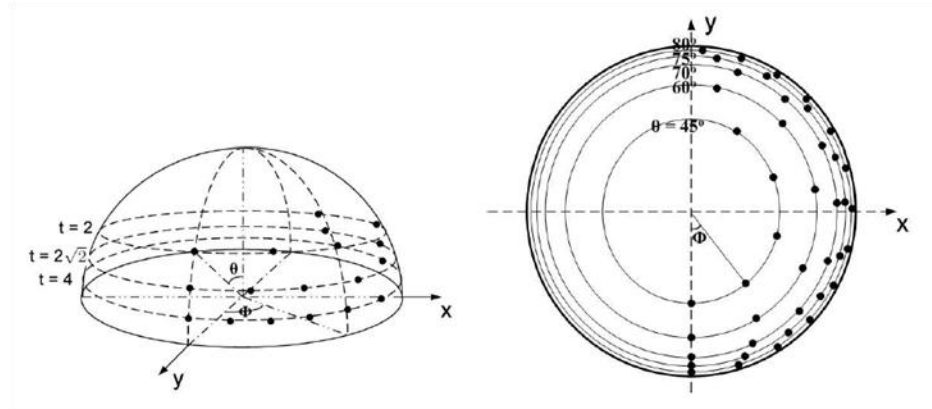


Figure 4 - 14 The sampling of the parameters $\theta = \arccos \frac{1}{t}$ and ϕ (Morel and Yu, 2009).

3. Lastly, implement SIFT (detailed in Section 4.1.2.1) to compute the features of all estimated images. Select the best pair with the largest amount correspondences.

4.3 Second Modification - Semi-Global Matching (SGM)

The point cloud from PMVS is too sparse. SGM is a better algorithm when dealing with photogrammetry problems (Gerke, 2009). It exploits the so-called epipolar constraint which reduces the search space for matches to a 1D problem.

4.3.1 The Workflow after the Second Modification

After the second modification, the workflow is changed as Figure 4-15 shows.

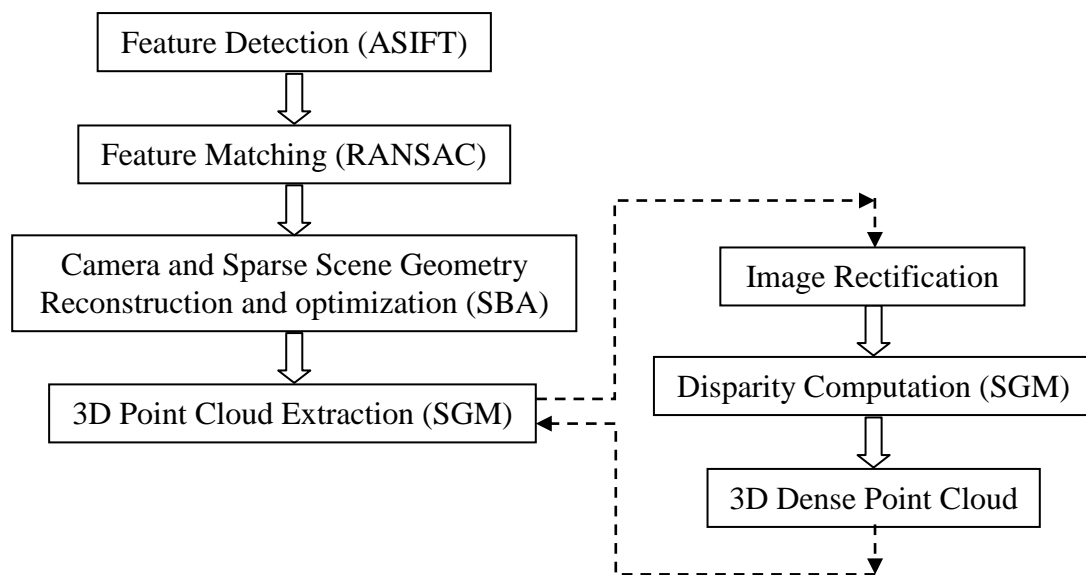


Figure 4 - 15 The workflow after the second modification using SGM algorithm for dense stereo matching

The modified workflow uses the camera parameters from SBA to rectify the images, and then computes the disparity map on the rectified images. At last, reproject each pixel in the image back to the 3D world based on the disparity map to realize a very dense 3D point cloud.

4.3.2 Image Rectification

Rectification is the essential step before applying the matching algorithm. In rectified images the epipolar lines are parallel to the x-axis of the image, which simplifies the epipolar constraint to the same line.

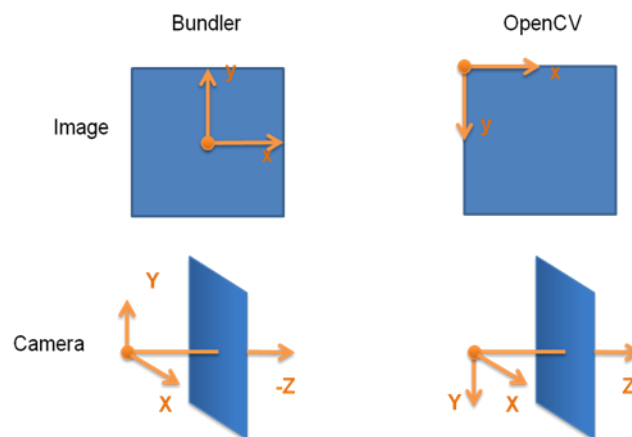


Figure 4 - 16 The comparison of different coordinate systems in Bundler and OpenCV.

In this project, a function named stereoRectify in an Open Source Computer Vision (OpenCV, 2013) library is used to rectify the images. It requires the internal and external matrices of cameras as the inputs, and fortunately, Bundler gives these parameters. However there are some differences between Bundler (Snavely, 2006) and OpenCV (Kolaric, 2007) coordinate systems, as Figure 4-16 shows.

Assume a pinhole camera model is used. In the Bundler coordinate system, a 3D world coordinate, \mathbf{X}_B^W , is projected by a camera with rotation matrix \mathbf{R}_B and translation matrix \mathbf{T}_B to a camera coordinate, \mathbf{X}_B^C :

$$\mathbf{X}_B^C = \mathbf{R}_B \mathbf{X}_B^W + \mathbf{T}_B. \quad (4-34)$$

Convert the world coordinate, \mathbf{X}_0^W , from OpenCV back to Bundler coordinate system:

$$\mathbf{X}_B^W = \mathbf{R}_w \mathbf{X}_0^W. \quad (4-35)$$

Here \mathbf{R}_w is the rotation matrix of world coordinate system from OpenCV to Bundler, which is 180 degree rotation around x-axis. And convert the camera coordinate, \mathbf{X}_0^C , from OpenCV back to Bundler coordinate system:

$$\mathbf{X}_B^C = \mathbf{R}_C \mathbf{X}_0^C. \quad (4-36)$$

Here, \mathbf{R}_C is the rotation matrix of camera coordinate system from OpenCV to Bundler, which is also 180 degree rotation around x-axis. Substituting \mathbf{X}_B^W and \mathbf{X}_B^C by equations (4-35) and (4-36), respectively, equation (4-34) becomes

$$\mathbf{R}_C \mathbf{X}_0^C = \mathbf{R}_B \mathbf{R}_w \mathbf{X}_0^W + \mathbf{T}_B. \quad (4-37)$$

Now, in the OpenCV coordinate system, the camera projective is described by

$$\mathbf{X}_0^C = \mathbf{R}_C^{-1} \mathbf{R}_B \mathbf{R}_w \mathbf{X}_0^W + \mathbf{R}_C^{-1} \mathbf{T}_B. \quad (4-38)$$

The new rotation matrix \mathbf{R}_0 and translation matrix \mathbf{T}_0 in OpenCV coordinates is

$$\mathbf{R}_0 = \mathbf{R}_C^{-1} \mathbf{R}_B \mathbf{R}_w, \quad (4-39)$$

$$\mathbf{T}_0 = \mathbf{R}_C^{-1} \mathbf{T}_B. \quad (4-40)$$

Then for function stereoRectify, the rotation matrix and translation matrix between the coordinate systems of the first and the second cameras are

$$\mathbf{R} = \mathbf{R}_2 \mathbf{R}_1^{-1} = \mathbf{R}_C^{-1} \mathbf{R}_{2B} \mathbf{R}_{1B}^{-1} \mathbf{R}_C \quad (4-41)$$

$$\mathbf{T} = \mathbf{T}_2 - \mathbf{R} \mathbf{T}_1 = \mathbf{R}_C^{-1} \mathbf{T}_{2B} - \mathbf{R}_C^{-1} \mathbf{R}_{2B} \mathbf{R}_{1B}^{-1} \mathbf{T}_{1B} \quad (4-42)$$

Since the original coordinate of image coordinate system of OpenCV is the top left corner, the principle point now is (w/2, h/2).

4.3.3 Semi-Global Matching (SGM) Algorithm

Dense stereo matching is often difficult due to occlusions, object boundaries, and low or repetitive textures. Scharstein and Szeliski (2002) separate most matching methods into four steps: matching cost computation, cost aggregation, disparity computation/optimization, and disparity refinement. Hirschmuller (2008) introduced a new way based on Mutual Information (MI) for handling complex radiometric relationships between images, to realize a pixelwise matching cost calculation. He proposed an approximate global, 2D smoothness constraint by combining many 1D constraints in the cost aggregation step.

4.3.3.1 Pixelwise Matching Cost Calculation

The matching cost is calculated for a pixel \mathbf{p} in the base image from its intensity I_{bp} and the suspected correspondence I_{mq} with $\mathbf{q} = e_{bm}(\mathbf{p}, d)$ in the match image. For rectified images, with the match images on the right of the base image, the epipolar line $e_{bm}(\mathbf{p}, d) = [p_x - d, p_y]^T$ with d as disparity.

The matching cost based on MI is derived from the entropy H of two images and their joint entropy, which are calculated from the probability distributions P of the intensities in the associated images.

$$MI_{I_1, I_2} = H_{I_1} + H_{I_2} - H_{I_1, I_2}, \quad (4-43)$$

$$H_I = - \int_0^1 P_I(i) \log P_I(i) di, \quad (4-44)$$

$$H_{I_1, I_2} = - \int_0^1 \int_0^1 P_{I_1, I_2}(i_1, i_2) \log P_{I_1, I_2}(i_1, i_2) di_1 di_2. \quad (4-45)$$

The better the images are registered, the lower the joint entropy H_{I_1, I_2} , the higher the value of MI.

In order to use pixelwise matching cost, the joint entropy H_{I_1, I_2} is expended via Taylor expansion into a sum over pixels (Kim et al., 2003). After warping the matching image according to the disparity image D by $f_D(I_m)$, the joint entropy H_{I_1, I_2} is

$$H_{I_1, I_2} = \sum_p h_{I_1, I_2}(I_{1p}, I_{2p}), \quad (4-46)$$

$$h_{I_1, I_2}(i, k) = -\frac{1}{n} \log \left(P_{I_1, I_2}(i, k) \otimes g(i, k) \right) \otimes g(i, k). \quad (4-47)$$

Here, P_{I_1, I_2} is the joint probability distribution of corresponding intensities, n is the total number of corresponding pixels, and $\otimes g(i, k)$ implies convolution with a 2D Gaussian with a small kernel (i.e. 7×7). Along the same expansion of joint entropy, the entropy of the two images is

$$H_I = \sum_p h_I(I_p), \quad (4-48)$$

$$h_I(i) = -\frac{1}{n} \log(P_I(i) \otimes g(i)) \otimes g(i), \quad (4-49)$$

with $P_{I_1}(i) = \sum_k P_{I_1, I_2}(i, k)$, $P_{I_2}(k) = \sum_i P_{I_1, I_2}(i, k)$. Then MI is

$$MI_{I_1, I_2} = \sum_p mi_{I_1, I_2}(I_{1p}, I_{2p}), \quad (4-50)$$

$$mi_{I_1, I_2}(i, k) = h_{I_1}(i) + h_{I_2}(k) - h_{I_1, I_2}(i, k). \quad (4-51)$$

A look-up table of MI is built up. So, the pixelwise matching cost based on MI is

$$C_{MI}(p, d) = -mi_{I_b, f_D(I_m)}(I_{bp}, I_{mq}). \quad (4-52)$$

Well registered images have high MI, which results in lower cost.

In this thesis, the cost was normalized to 32bit float that the max position becomes 1.0 and the min position becomes 0.0, suggested by Matthias Heinrichs (2007).

A disparity map is required for warping I_m before probability calculation. Kim et al. (2003) suggested an iterative solution that a final disparity map could be calculated after a rather low (e.g. 3) number of iterations with a random disparity map as a start. This is because that even a

wrong disparity map could give a good estimation of the probability distribution P when the number of pixels is high enough. However, the computational time would be very high unnecessarily. Hirschmuller (2008) proposed a fast way through a hierarchical calculation. He suggested to run three iterations with a random disparity map as the initial disparity at a resolution of $1/16$, and then recursively use the (up-scaled) disparity image calculated at half resolution as the initial disparity, backing to the original resolution gradually. Theoretically, the runtime would be just 14 percent slower than the runtime of one iteration at the original resolution, ignoring the overhead of MI calculation and image scaling.

4.3.3.2 Cost Aggregation

Due to noise, an additional constraint is needed to smooth and penalize changes of neighboring disparity. The energy $E(D)$ depending on the disparity map D is

$$E(D) = \sum_p \left(C(p, D_p) + \sum_{q \in N_p} P_1 T[|D_p - D_q| = 1] + \sum_{q \in N_p} P_2 T[|D_p - D_q| > 1] \right), \quad (4-53)$$

where the operator $T[]$ is the probability distribution of corresponding intensities. It is 1 if its argument is true and 0 otherwise.

The first term of Equation (4-53) is the sum of matching costs of all pixels for the disparity map D . The second term adds a small constraint for all pixels q within the neighborhood of p if the disparity changes 1 pixel, and the third term adds a larger penalty for the larger disparity changes, ensuring that $P_2 \geq P_1$. In this thesis, $P_1 = 0.05$, and $P_2 = [0.06, 0.8]$ depending on the intensity gradient in the original image (Heinrichs, 2007).

Finding the disparity map D that minimizes the energy $E(D)$ is a global minimization in 2D. It is a NP-complete problem. Hirschmuller (2008) divided the 2D aggregation into 1D from all direction equally. Figure 4-17 shows the calculation of the aggregated cost $S(\mathbf{p}, d)$ for a pixel \mathbf{p} and disparity d . It summarizes the costs of all 1D minimum cost paths ending in pixel \mathbf{p} at disparity d .

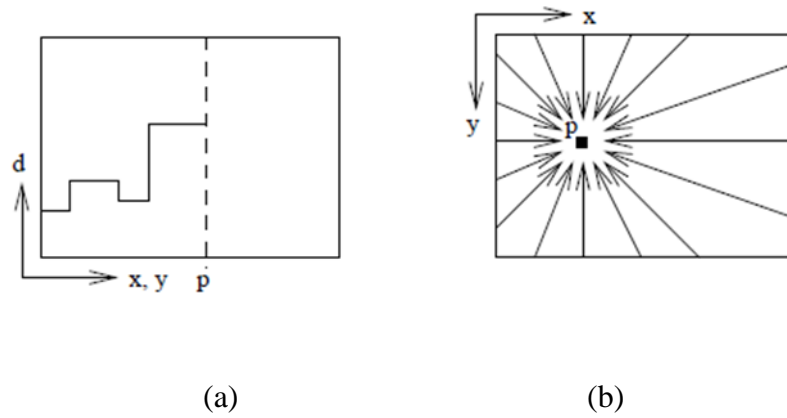


Figure 4 - 17 The aggregation of cost through 16 paths. (a) is the minimum cost path $L_r(\mathbf{p}, d)$, (b) is the 16 paths from all directions \mathbf{r} for pixel \mathbf{p} (Hirschmuller, 2008).

The cost $L_r(\mathbf{p}, d)$ along a path traversed in the direction \mathbf{r} of the pixel \mathbf{p} at disparity d is defined recursively as

$$L_r(\mathbf{p}, d) = C(\mathbf{p}, d)$$

$$+ \min \left(L_r(\mathbf{p} - \mathbf{r}, d), L_r(\mathbf{p} - \mathbf{r}, d - 1) + P_1, L_r(\mathbf{p} - \mathbf{r}, d + 1) + P_1, \min_i L_r(\mathbf{p} - \mathbf{r}, i) + P_2 \right) - \min_k L_r(\mathbf{p} - \mathbf{r}, k). \quad (4-54)$$

The last term is the minimum path cost of the previous pixel from the whole term. This will not change the actual aggregated cost but limits the upper value of $L_r(\mathbf{p}, d)$ as well as $S(\mathbf{p}, d)$. Summarizing the costs L_r over at least 8 (and should be 16) paths \mathbf{r} provides good coverage of the 2D image, as

$$S(\mathbf{p}, d) = \sum_r L_r(\mathbf{p}, d). \quad (4-55)$$

The 8 paths are horizontal, vertical and diagonal, while the 16 paths are 8 paths adding one step horizontal, one step vertical and one step diagonal.

4.3.3.3 Disparity Computation

For each pixel \mathbf{p} , select the disparity d that corresponds to the minimum sum of cost. So the disparity map corresponding to the base image I_b is $D_b = \min_d S_b[\mathbf{p}, d]$. Switch the roles of base image and match image, we can get $D_m = \min_d S_m[e_{mb}(\mathbf{q}, d), d]$. In order to obtain sub-pixel estimation, a parabolic curve was fitted to minimum, the next higher and lower disparity, and the position of the minimum of the curve is calculated as the sub-pixel matching disparity.

After filtering both disparity maps by a median filter with a small window, i.e. 3×3 , to remove outliers, the disparity map is determined by a consistency check

$$D_p = \begin{cases} D_{bp} & \text{if } |D_{bp} - D_{mq}| \leq 1 \\ D_{inv} & \text{otherwise} \end{cases}, \quad (4-56)$$

with $\mathbf{q} = e_{bm}(\mathbf{p}, D_{bp})$. This consistency check ensures unique mappings.

4.3.3.4 The Algorithm Steps

The SGM algorithm's workflow in the work done by Hirschmuller (2008) is as Figure 4-18 shows.

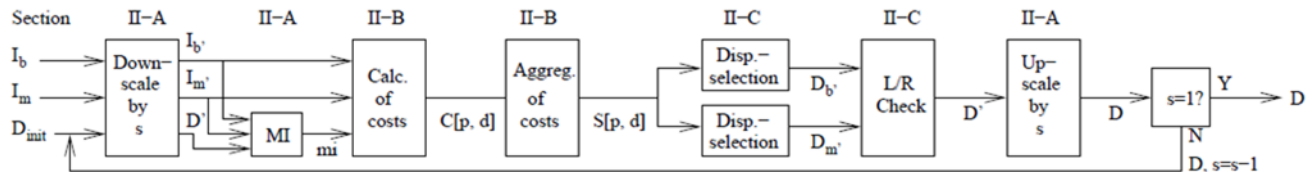


Figure 4 - 18 The workflow of SGM algorithm (Hirschmuller, 2008).

Here are some key steps for coding hinted by Heinrichs (2007):

- Building the Probability map: Initialize a 256×256 array $P(I_b, I_m)$ with the value of ONE (this helps avoid $\log(0)$ later on). Checking each pixel p in the base image, if it has a valid correspondence q in the match image with a knowing disparity (a random number in the first iteration), increase $P(I_{bp}, I_{mq})$ by one. After going through all the pixels, the probability map is the array P divided by the total number of P .
- Compute the entropies according to Equation (4-46) and (4-48).
- Build up a look-up table for MI according to Equation (4-50) and normalize it.
- Determining the Cost Matrix: Initiate a $M \times N \times D$ matrix $C(p, q, d)$ if the images are the size of $M \times N$ and the disparity range is D . Checking each pixel p in the base image with a

valid correspondence q in the match image when disparity is d , give $C(p, q, d)$ the value $MI(I_{bp}, I_{mq})$ from the MI look-up table.

- e) Cost aggregation: Take from left to right horizontally as an example. Initial a matrix S with the same size as cost matrix C with all 0 values for storing the sum of cost, and a buffer $\min C$ with the same size as the height of the image for storing the minimum costs of the certain rows. For each row in the first column, copy the value from the cost matrix C to the matrix S for all disparity values, and store the minimum cost of each pixel in the buffer $\min C$. On the next column, aggregate the cost as Equation (4-53) does and replace the minimum cost at the end. Do so for all columns. After finishing the first path, repeat this cost aggregation steps to all the other 15 paths: horizontal, vertical, diagonal and one step horizontal or vertical and one step diagonal.
- f) Determining disparity d_{\min} for each pixel with minimum value in the sum of cost matrix S . Compute the minimum disparity on a parabolic curve fitted by minimum, the next higher and lower values. And then filtered the disparity map by a medium filter. This is the disparity map D_b corresponding to the base image.
- g) Alternate the role of the base image and the match image and repeat steps from a) to f) to calculate the disparity map D_m corresponding to the match image.
- h) Check the consistency via Equation (4-55).

4.3.3.5 Disparity Refinement

The disparity map may still have some errors and invalid values. So disparity refinement steps are needed. These include removal of peaks and discontinuity preserving interpolation.

For peak removal, the disparity map is segmented by allowing one pixel varying within one segment, considering 4 connected neighbors. The disparity segments with the size smaller than a threshold are set to invalid.

Due to consistency check or peak filtering, some pixels on the disparity map are set to invalid. In order to interpolate the holes while at the same time preserve the discontinuity, the invalid disparities are classified into two classes: occlusions and mismatches first. Then interpolate the occlusions by the background, while interpolate the mismatches by all neighboring pixels.

For each invalid disparity d_{ik} , find the nearest valid disparities above d_{ia} , below d_{ib} , left d_{lk} and right d_{rk} , and their correspondent segment indexes S_{ik} , S_{ia} , S_{ib} , S_{lk} , S_{rk} . The disparity d_{ik} is then determined by the following interpolation equations (Hirschmuller, 2003).

$$d_{ik} = \begin{cases} \frac{d_h + d_v}{2} & \text{if } S_h = S_v \\ \min(d_h, d_v) & \text{if } S_h \neq S_v \end{cases} \quad (4-57)$$

with

$$d_h = \begin{cases} \frac{(d_{rk} - d_{lk})(i-1)}{r-1} + d_{lk} & \text{if } S_{lk} = S_{rk} \\ \min(d_{lk}, d_{rk}) & \text{if } S_{lk} \neq S_{rk} \end{cases} \quad (4-57a)$$

$$d_v = \begin{cases} \frac{(d_{ib} - d_{ia})(k-a)}{b-a} + d_{ia} & \text{if } S_{it} = S_{ib}, \\ \min(d_{ia}, d_{ib}) & \text{if } S_{ia} \neq S_{ib}. \end{cases} \quad (4-57b)$$

$$S_h = \begin{cases} S_{lk} & \text{if } d_{lk} < d_{rk}, \\ S_{rk} & \text{if } d_{lk} \geq d_{rk}. \end{cases} \quad (4-57c)$$

$$S_v = \begin{cases} S_{ia} & \text{if } d_{ia} < d_{ib}, \\ S_{ib} & \text{if } d_{ia} \geq d_{ib}. \end{cases} \quad (4-57d)$$

4.3.3.6 Processing of Large Images

Due to the large temporary memory requirement for storing the pixelwise matching cost matrix and aggregated cost matrix, larger images, such as the Pictometry images, should be divided into several tiles. The tiles should have overlapping areas to avoid mismatches near tile borders. In this thesis, the tile size was chose as 1000 rows by image width, and the overlap is 200 rows. After computing all the tiles, a weighted mean of disparities from all tiles at overlapping areas is calculated, shown as Figure 4-19.

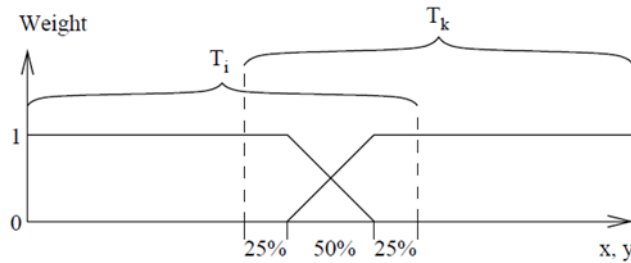


Figure 4 - 19 Merging all tiles by calculating a weighted mean at overlapping areas

(Hirschmuller, 2008).

In the left 25% of the overlapping area of tile T_i and tile T_k , only the disparity from T_i is considered. In the middle 50% of the overlapping area, a weighted mean of disparities from both tiles is calculated. In the right 25% of the overlapping area, only the disparity from T_k is considered. In this thesis, the overlap area is 200 pixels, and only center 100 pixels are considered for disparity merging.

4.3.4 Dense Point Cloud Projection from Disparity Map

Stereo vision recovers object's 3D information based on disparity and triangulation (Zou and Li, 2010). Once the disparity map has been computed by stereo matching, basic photogrammetry (as Figure 4-20 shows) can be used to calculate the 3D coordinates for each valid pixel, whose value is larger than 0 in the disparity map.

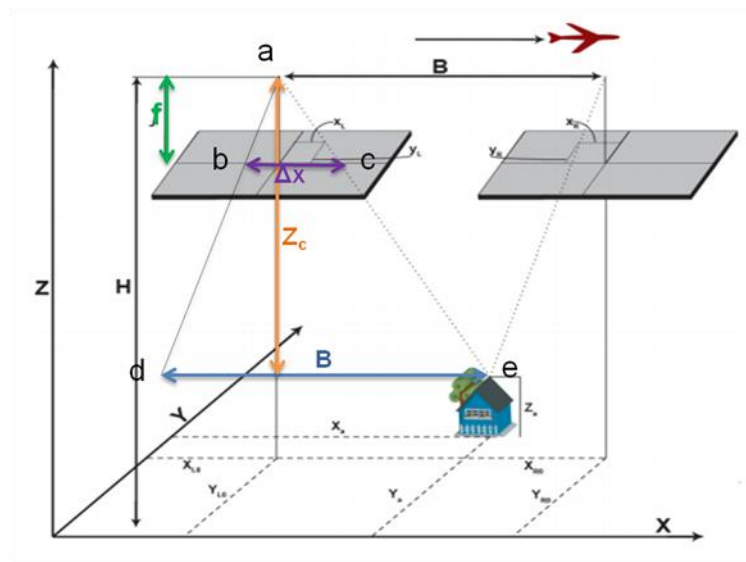


Figure 4 - 20 The model used for calculating the 3D coordinates in basic photogrammetry

(Nilosek and Salvaggio, 2010).

In Figure 4-20, B is the baseline between the two images, f is the focal length, the subscripts l and r refer to the left and right images, respectively (Nilosek and Salvaggio, 2010), and Z_c is the distance of 3D point away from the camera. If moving the right part to the left, Δx represents the x displacement of the stereo image pair. Then through the similarity of triangles: triangle abc is similar to triangle ade, the coordinate, Z_c , can be calculated by

$$Z_c = \frac{f \cdot B}{\Delta x}. \quad (4-58)$$

Zou and Li (2010) used a reproject matrix, Q, from the output of rectification function in OpenCV to compute 3D coordinates. Q matrix, gives all the photogrammetry information needed for reprojection. If the stereo images were placed parallel to the x axis, the Q matrix is

$$Q = \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{c_x - c'_x}{T_x} \end{pmatrix}, \quad (4-59)$$

where, c_x and c_y are the principal point coordinates in the x and y axis, respectively, in the left image after rectification, c'_x is the x principal point coordinate in the right image, f is the focal length of the left camera, and T_x is the baseline of stereo images.

For a valid pixel (x, y) with disparity d, the homogeneous coordinates (X, Y, Z, W) for 3D point are computed by

$$\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = Q \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} x - c_x \\ y - c_y \\ f \\ \frac{-d + c_x - c'_x}{T_x} \end{pmatrix}. \quad (4-60)$$

Finally, the object coordinates are decided as X/W , Y/W , Z/W . They are

$$X = \frac{T_x(x - c_x)}{-d + c_x - c'_x}, \quad (4-61)$$

$$Y = \frac{T_x(y - c_y)}{-d + c_x - c'_x}, \quad (4-62)$$

$$Z = \frac{T_x f}{-d + c_x - c'_x}. \quad (4-63)$$

4.4 Third Modification - Noise Removal

There will be some noise definitely on the point cloud due to errors in the keypoints detected by SIFT, wrong matching by RANSAC, errors arising in the SBA optimization step, or incorrect disparity computed by SGM. These outliers will complicate the estimation of local point cloud characteristics such as surface normals or curvature changes, which may cause point cloud registration failures. So noise removal is an essential step after point cloud extraction. In this thesis, two noise removal methods were tested and compared in the point cloud space. They are statistical removal and radius outlier removal, provided by the Point Cloud Library (PCL, 2013). And for the SGM only, another noise removal method, bilateral filter, was tested on the disparity map.

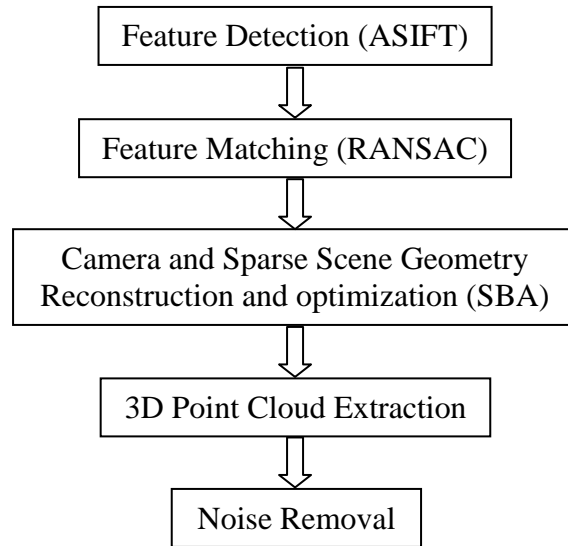


Figure 4 - 21 The workflow after the third modification with noise removal.

4.4.1 The Workflow after the Third Modification

After the third modification, the workflow is as Figure 4-21 shows. In the modified workflow, a noise removal step is added at last to remove extraneous points from the extracted dense point cloud.

4.4.2 Noise Removal Methods

4.4.2.1 Statistical Removal

The statistical removal method performs a statistical analysis on a neighborhood of each point. It assumes that the distribution of point-to-neighbor distances in the input dataset is a Gaussian distribution with a mean and a standard deviation. For each point, the mean distance from it to all

its neighbors is calculated. The point would be trimmed from the dataset if the mean distance is outside the threshold defined the standard deviation.

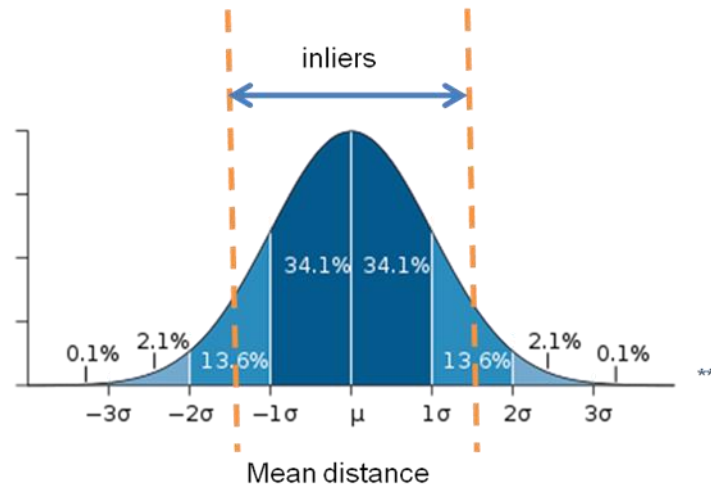


Figure 4 - 22 The diagram of the statistical removal method (Wikipedia).

Figure 4-22 shows the diagram of Statistical Removal. In Figure 4-22, the mean distances of all points to their neighbors fall in a Gaussian distribution, as blue area shows. Then set a threshold shown as the orange dash line. The points with the mean distance within the threshold are inliers, others are outliers.

4.4.2.2 Radius Outlier Removal

The radius outlier removal method counts the number of points within a certain radius of a given point. If it is lower than some threshold, that point would be removed as an outlier, shown as in Figure 4-23.

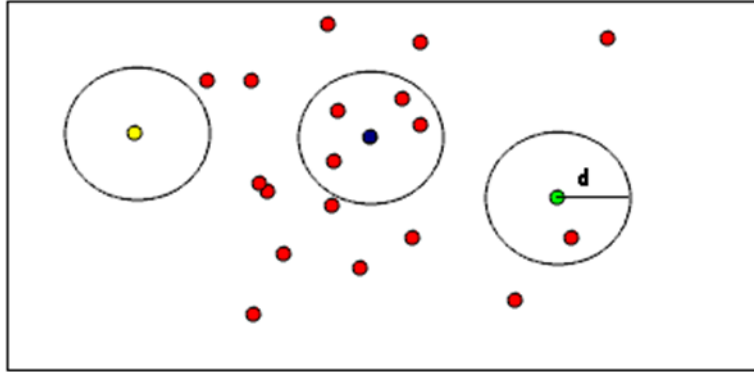


Figure 4 - 23 The diagram of the radius outlier removal method (PCL, 2013).

In Figure 4-23, a circle (a spherical in 3D point cloud) is set to each point with radius d and the center at that point. Count the points within the circle (the spherical in 3D point cloud), such as zero for the yellow point, four for the blue point, and one for the green point. If the threshold is 2, then the yellow point and blue point are outliers while the blue point is an inlier.

4.4.2.3 Bilateral Filter

A bilateral filter is an edge-preserving and noise reducing smoothing filter. The average weights of this filter combine geometric closeness and photometric similarity from nearby pixels of each pixel in the processing image. Here, bilateral filter was applied to the disparity map, and the photometric similarity is the disparity closeness.

A lowpass spatial domain filter applied to a multiband image $f(x)$ produces a multiband response image as follows

$$h(x) = \frac{1}{k_d(x)} \iint_{-\infty}^{\infty} f(\xi) c(\xi) d\xi \quad (4-64)$$

where $c(\xi, x)$ is a measure of geometric closeness between the neighborhood centre x and a nearby point ξ . The normalization factor $k_d(x)$ is given by

$$k_d(x) = \iint_{-\infty}^{\infty} c(\xi, x) d\xi \quad (4-65)$$

The range (brightness) domain filtering is carried out similarly as

$$h(x) = \frac{1}{k_s(x)} \iint_{-\infty}^{\infty} f(\xi) s(f(\xi), f(x)) d\xi \quad (4-66)$$

where $s(f(\xi), f(x))$ measures the photometric (brightness) similarity between the pixel at the neighborhood centre and that of a nearby point ξ . The normalization factor in this case is

$$k_s(x) = \iint_{-\infty}^{\infty} s(f(\xi), f(x)) d\xi \quad (4-67)$$

Bilateral filtering simultaneously combines the spatial and range domain filters

$$h(x) = \frac{1}{k(x)} \iint_{-\infty}^{\infty} f(\xi) c(\xi, x) s(f(\xi), f(x)) d\xi \quad (4-68)$$

where the normalization factor is

$$k(x) = \iint_{-\infty}^{\infty} c(\xi, x) s(f(\xi), f(x)) d\xi \quad (4-69)$$

Bilateral filtering replaces the pixel value at x with an average of similar and nearby pixel values. In smooth regions, pixel values in a small neighborhood are similar to each other and the

bilateral filter averages away the small differences between pixel values caused by noise. For sharp edges, good filtering behavior is achieved at the boundaries due to the closeness component, and edges are preserved at the same time due to the range component.

Gaussian filter is the most common implementation of bilateral filtering. It is applied for both the closeness, $c(\xi, x)$, and the similarity, $s(f(\xi), f(x))$ functions. For the closeness function we have

$$c(\xi, x) = e^{-\frac{1}{2} \left(\frac{d(\xi, x)}{\sigma_d} \right)^2} \quad (4-70)$$

where $d(\xi, x)$ is the Euclidian distance between x and ξ

$$d(\xi, x) = d(\xi - x) = \| \xi - x \| \quad (4-71)$$

Analogously we have the similarity function

$$s(\xi, x) = e^{-\frac{1}{2} \left(\frac{\delta(f(\xi), f(x))}{\sigma_r} \right)^2} \quad (4-72)$$

where $\delta(f(\xi), f(x))$ is the Euclidean distance between two intensity values $f(\xi)$ and $f(x)$, namely

$$\delta(f(\xi), f(x)) = \delta(f(\xi) - f(x)) = \| f(\xi) - f(x) \| \quad (4-73)$$

which in the disparity image case simply involve disparity values.

The standard deviation, σ_d in the closeness filtering is chosen based on the desired amount of low-pass filtering. A large σ_d covers more distant image locations, thus blurs more. The standard

deviation, σ_r in the range filtering is chosen to achieve the desired amount of combination of pixel values. Pixels with value difference smaller than σ_r are mixed together.

5 Results

5.1 Results of the Previous Work

In order to reduce the execution time during initial testing, a small region around the Carlson building of RIT was cut out from the original Pictometry images. The size of the small region is 1000×1000 pixels (shown as Figure 5-1(a)), out of 3248×4872 pixels in the original images. Figure 5-1(b) gives the result of ten North-viewing oblique images (as Figure 3-2(a) shows) from the RIT 3D workflow.

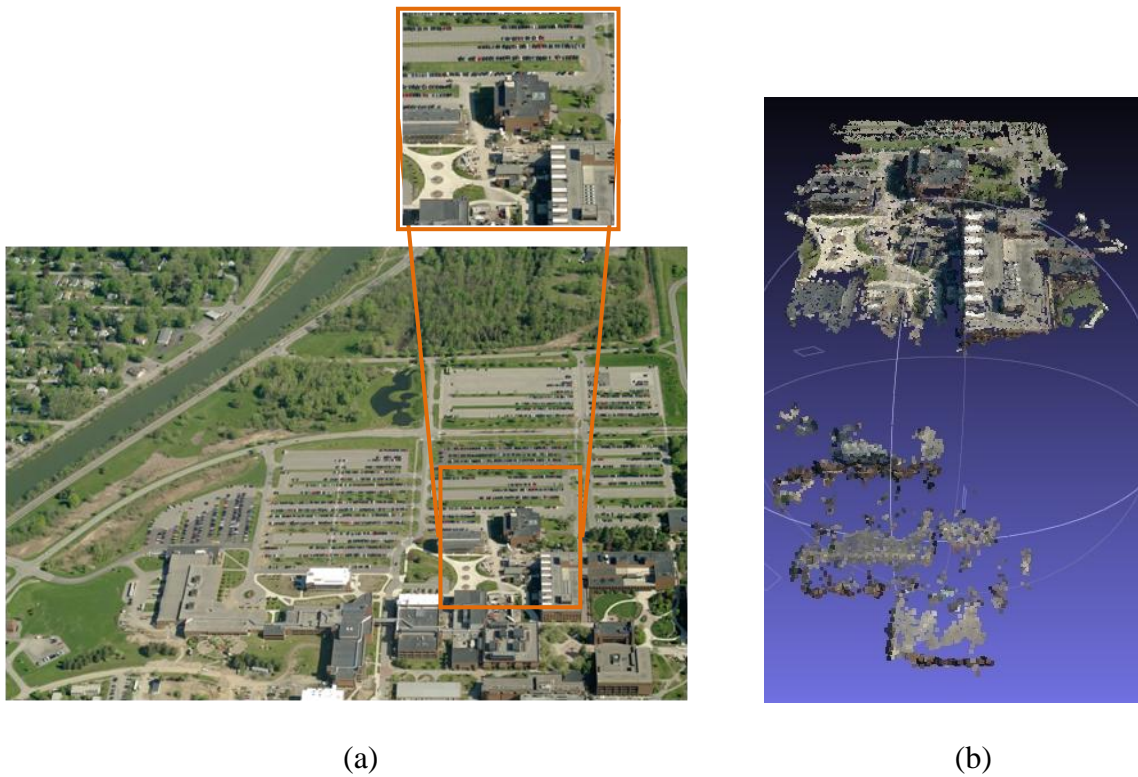


Figure 5-1 (a) is the small region around Imaging Science Building in the size of 1000×1000 pixels. (b) is the point cloud of 10 North-viewing small region images

The RIT 3D workflow gives a good result, 34,358 vertices in total. On the visible part from North-viewing images, a sparse point cloud of building roofs and walls are presented.

5.2 Results of the First Modification – ASIFT

ASIFT is an affine invariant extension of SIFT. It finds out many more keypoints and then results in more matches in oblique images when there is large deformation in the images viewing the same scene. As compared in Figure 5-2, ASIFT gives 85 matches while SIFT finds nothing when the deformation is as Figure 5-2(a) shows.



Figure 5-2 The comparison of ASIFT and SIFT to find the correspondences in oblique images. (a) is the image pair. (b) is the correspondences computed by ASIFT and SIFT: left of (b) is the result of ASIFT, right of (b) is the result of SIFT.

After modifying the first part of the workflow from SIFT to ASIFT, the same ten small regions of North-viewing images were processed by the modified workflow. Figure 5-3 compares the results. The modified ASIFT workflow contains 44,640 vertices, almost 30% more than the vertices from the SIFT workflow. This better result demonstrates that ASIFT is a better algorithm in finding keypoints with higher accuracy.

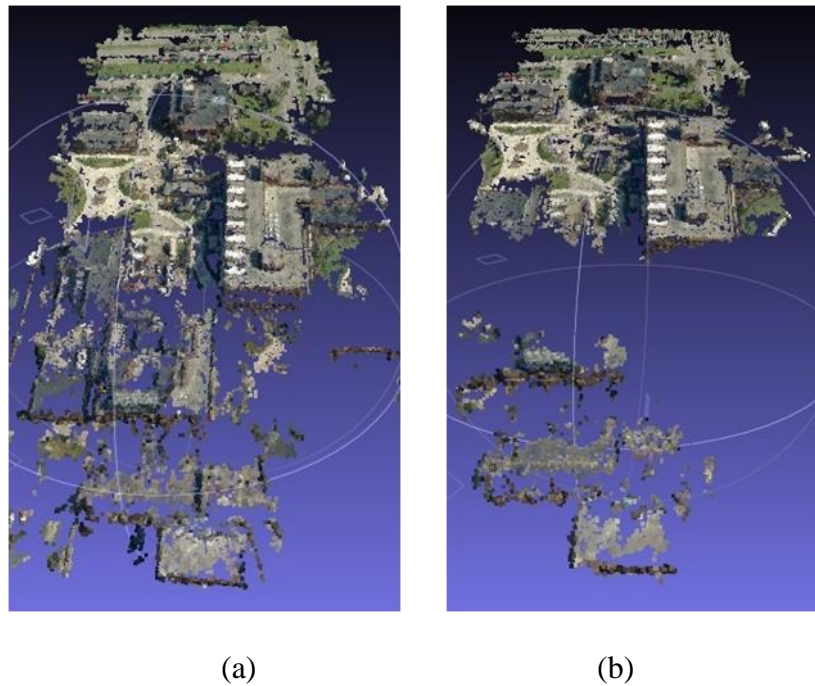


Figure 5-3 The point cloud of ten small regions of North-viewing Pictometry images. (a) is the result of ASIFT workflow (44640 vertices), (b) is the result of SIFT workflow (34358 vertices).

5.2.1 Results of Original Size Images

However, if you zoom in to check each wall on the buildings, no matter the point cloud from SIFT or ASIFT workflow, you will find out they are not vertical to the ground (shown as Figure

5-4). The reason is that Bundler assumes the principle point is in the center of each image. So image cutting gives the system wrong coordinates of principle point, and furthermore wrong estimation of the camera parameters, which fools the PMVS in computing the 3D point clouds.

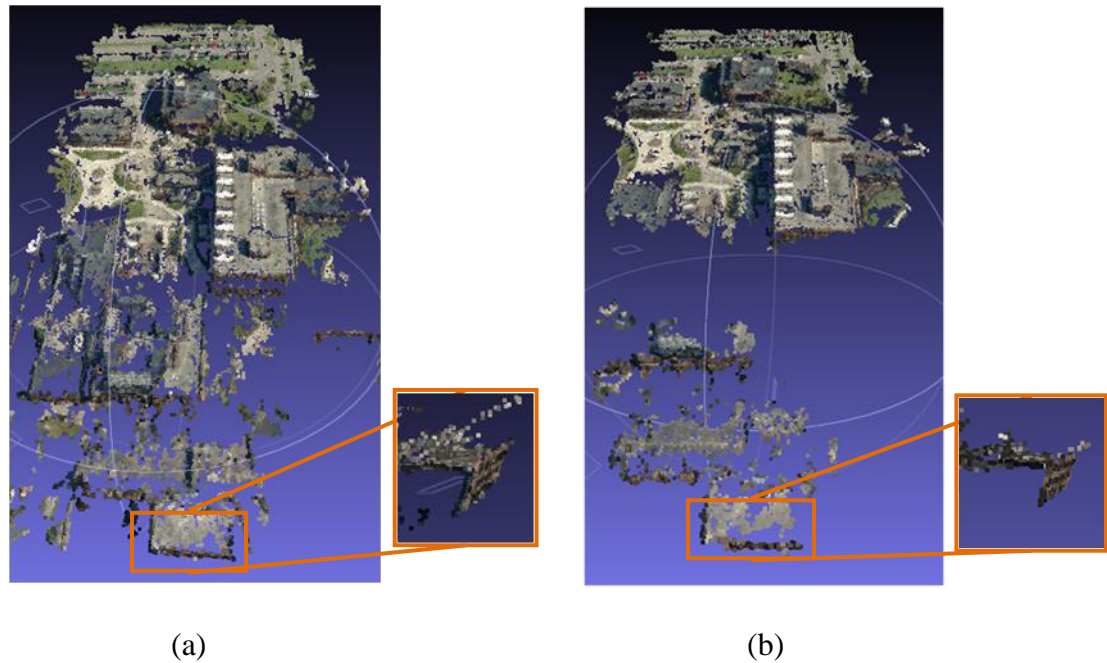


Figure 5-4 The errors on the walls in both point clouds. (a) is the point cloud from ASIFT workflow, (b) is the point cloud form SIFT workflow

Using the original images, the point clouds from the same ten original Pictometry oblique images are shown in Figure 5-5. The point cloud covers more regions because the original images cover a larger region. Comparison of the results from ASIFT workflow and from SIFT workflow reveals that the point cloud from ASIFT workflow covers more regions, especially in the top right of the figures, and also contains 40% more vertices than the SIFT workflow. These results confirm again that ASIFT is better than SIFT in dealing with the oblique images.

From the zoom image in Figure 5-6, all the walls are vertical to the ground. After going back to the original size, it minimizes the error brought by the error estimation of the principle point in Bundler.

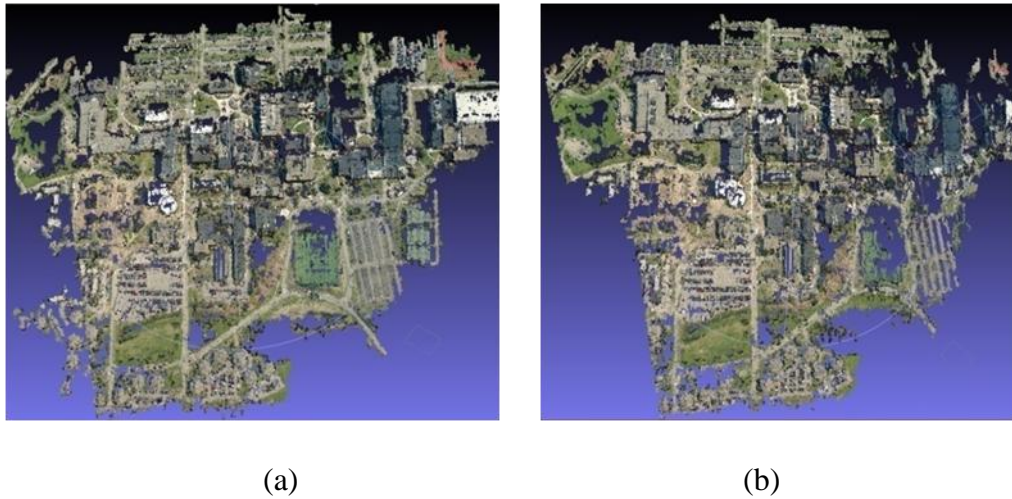


Figure 5-5 The point clouds of original size images. (a) is the result from ASIFT workflow (506,084 vertices), (b) is the result from SIFT workflow (357,702 vertices).

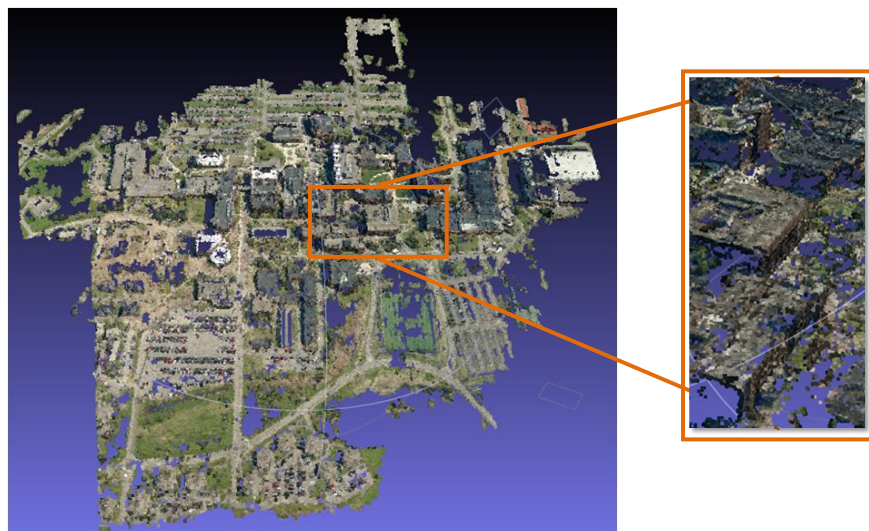


Figure 5-6 The vertical walls after going back to the original size images.

5.2.2 Results of Focal Length Fixation

Although the walls are now vertical, there still are some fatal errors on the details of the walls for either the SIFT or ASIFT result. These errors are part of the walls floating outside the building, marked as red points in Figure 5-7.



Figure 5-7 The floating walls marked by red points.

The walls are floating because of the optimization of SBA. Although SBA could compute and optimize the camera parameters, it does not guarantee to give the correct focal lengths. Table 5-1 shows the difference of focal lengths after optimization compared to the exact ones, which are also presented in Figure 5-8. The small red circles represent input focal lengths in Bundler, while the green circles represent the output focal lengths. Obviously, the trend of output focal lengths is similar to the actual focal lengths (the small blue circles), but they are not identical. The difference would bring errors on the walls.

Table 5-1 Focal lengths for the ten North-viewing images after SBA.

Images	Actual F	Input F	Output F
N1	11423	11423	11505
N2	11434		11439
N3	11423		11579
N4	11473		11496
N5	11483		11563
N6	11423		11522
N7	11473		11530
N8	11483		11571
N9	11434		11475
N10	11423		11530

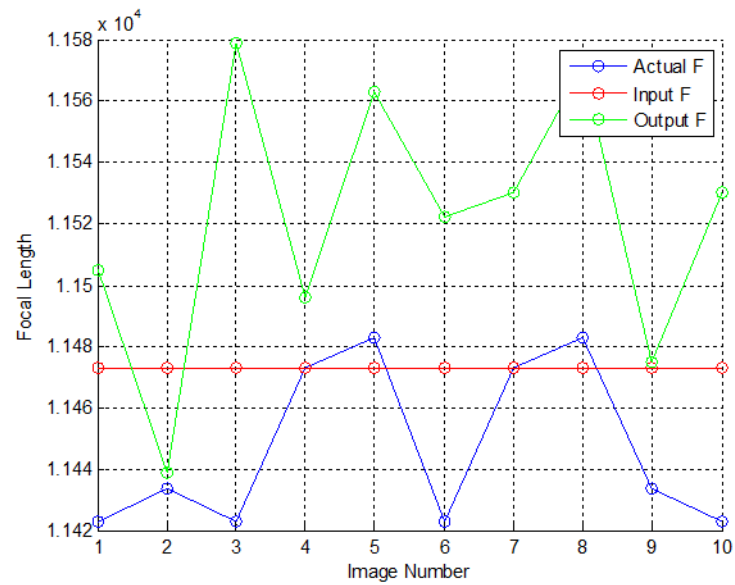






Figure 5-8 The diagram of the input and output focal lengths of Bundler, comparing to the actual focal length.

Table 5-2 Different output focal lengths with different values for constrain_focal_weight.

Images	Actual F	Input F	Output F of Diff. constrain_focal_weight			
			1.0e-4	1.0e6	1.0e12	1.0e24
N1	11423	11423	11506	11423	11423	11423
N2	11434	11434	11441	11434	11434	11434
N3	11423	11423	11579	11503	11502	11423
N4	11473	11473	11479	11473	11473	11473
N5	11483	11483	11563	11483	11483	11483
N6	11423	11423	11521	11423	11423	11423
N7	11473	11473	1512	11473	11473	11473
N8	11483	11483	11570	11483	11483	11483
N9	11434	11434	11473	11434	11434	11434
N10	11423	11423	11527	11423	11423	11423

Table 5-3 The vertices decrease when the value of constrain_focal_weight goes up.

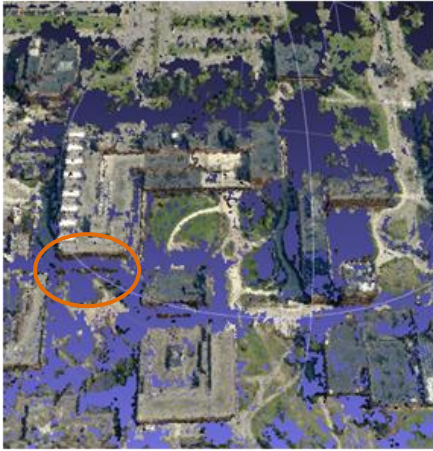
constrain_focal_weight			
1.0e-4	1.0e6	1.0e12	1.0e24
			
506,774 vertices	517,026 vertices	514,334 vertices	440,595 vertices



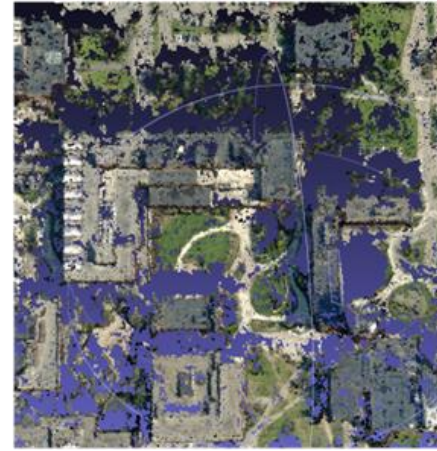
(a)



(b)



(c)



(d)

Figure 5-9 Wall errors decrease when the value of parameter constrain_focal_weight increases. (a), (b), (c) and (d) are the point clouds of constrain_focal_weight with the value of $1.0e-4$, $1.0e6$, $1.0e12$ and $1.0e24$, respectively.

One solution for removing these floating walls is to use the actual focal length as the input focal length and constraining the variance of the focal length during the optimization process.

Fortunately, there is a constraint parameter in Bundler, named `constrain_focal_weight`. The greater the value of this parameter, the harder for Bundler to change the focal lengths when optimizing the camera parameters and 3D coordinates. After giving the Bundler the actual focal length for each image, with different values for `constrain_focal_weight`, the output focal lengths are shown in Table 5-2.

When the value of `constrain_focal_weight` goes up, the output focal lengths are getting more and more close to the actual ones. They are identical to the actual focal lengths when the parameter `constrain_focal_weight` equals to $1.0e24$. Although the total vertices of the point clouds decrease when `constrain_focal_weight` increase, as Table 5-3 shows, the errors on the walls of buildings are gradually corrected as the focal lengths getting closer to the actual ones, as Figure 5-9 shows. In Figure 5-9, the floating walls are marked by orange ellipses. When `constrain_focal_weight` reaches the value of $1.0e24$, no floating walls are in the point cloud.

5.3 Results of the Second Modification – SGM

The ASIFT workflow gives us a good point cloud. However, it is insufficient for finding features, like 3D keypoints for 3D registration, or building roof borders for modeling. As Figure 5-10 shows, the points are insufficient for roof extraction, since the borders are unclear enough.



Figure 5-10 Sparse point cloud extracted by ASIFT workflow.

So a much more dense point cloud is desired for the future research. In this project, SGM is tested for extracting a much more dense point cloud from oblique images. Before SGM implementation, the images should be rectified.

5.3.1 Results of Image Rectification

Converting the camera parameters, like rotation and translation matrix, from SBA coordinate system to OpenCV coordinate system, the rectification result of two North-viewing Pictometry images (shown as Figure 5-11(a) and (b)) are shown in Figure 5-11(c).

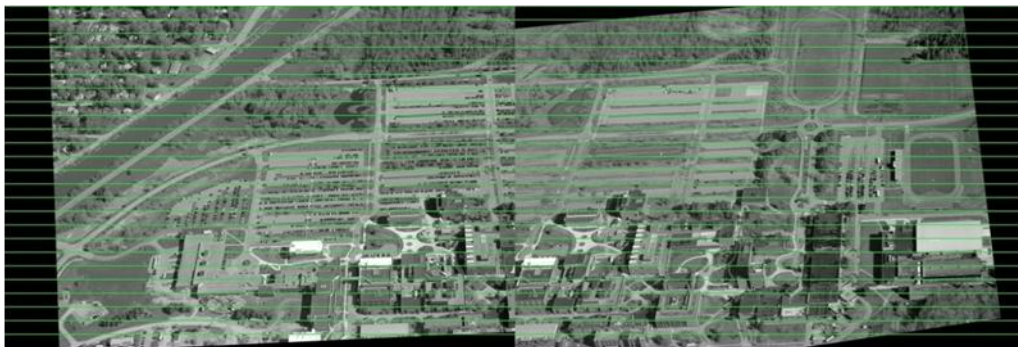
After rectification, the epipolar lines are parallel to the x-axis of the images. The same features on different images of the stereo pair now have the same y coordinates. This result reduces the search space for matches to a 1D problem. For the same features, it only needs to consider the shift of x- axis. The closer the distance from the feature to the camera, the larger the shift in x- axis. As Figure 5-12 demonstrates, the blue dot is closer to the camera than the orange dot and has larger x shift.



(a)



(b)



(c)

Figure 5-11 Image rectification result. (a) and (b) is the left and right images for rectification. (c) is the rectification result.

5.3.2 Results of Calculation of Disparity Maps

A function named SGBM (Semi-Global Block Matching) embedded in OpenCV is chosen to compute the disparity map from rectified image pairs. It is a simple version of SGM, and computes the disparity map in a block size. Figure 5-13 demonstrates the result from the Tsukuba images.



Figure 5-12 The diagram of disparity shift of features with different distance to the camera.

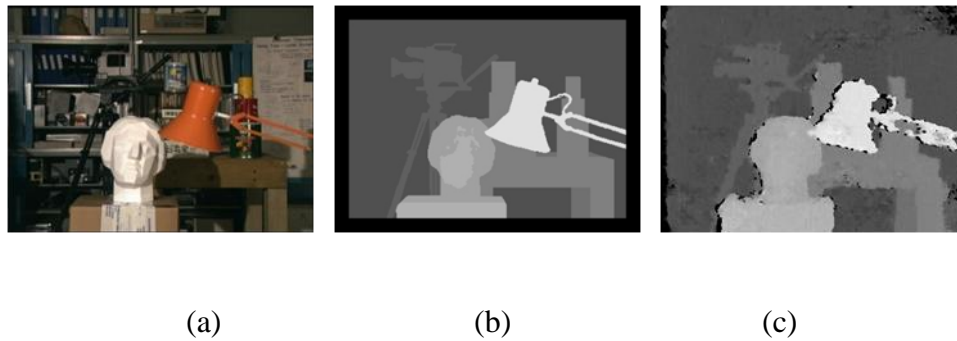


Figure 5-13 The disparity maps of Tsukuba images by SGBM function embedded in OpenCV. (a) is the image, (b) is the ground truth, (c) is the disparity map.

Comparing to the ground truth, the result mainly reveals the depth information well, like the lamp is brightest of all because it is closest to the camera, and the shelf is darkest due to it is

farthest from the camera. That depth information also shows in the disparity map computed from the rectified toys images (Figure 5-14).

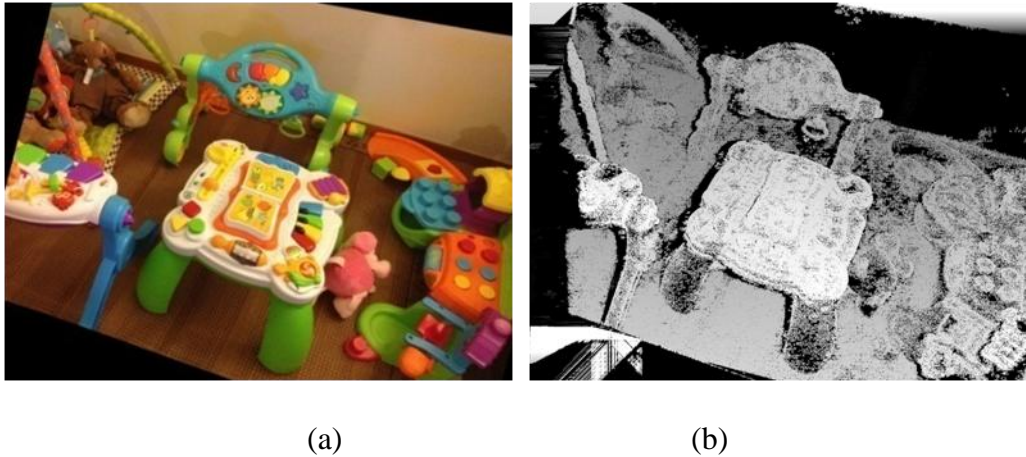


Figure 5-14 The disparity maps of toys images by SGBM function embedded in OpenCV. (a) is the images after rectification, (b) is the disparity map.

The disparity map of one pair of Pictometry oblique images, computed by SGBM in OpenCV, shows in Figure 5-15. We also can easily see the depth information from it.

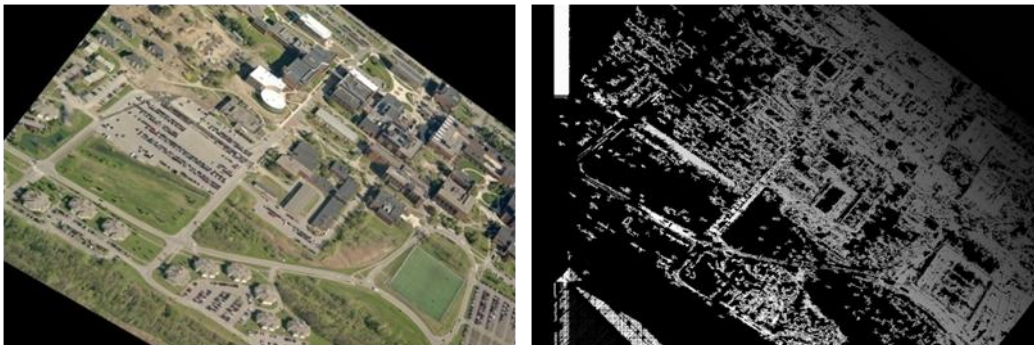


Figure 5-15 The disparity map of Pictometry oblique images computed by SGBM in OpenCV.

5.3.3 Results of Dense Point Cloud from Disparity Maps

The similarity of triangles in basic photogrammetry is used to compute the 3D coordinates of each valid pixel. Figure 5-16 gives the results of dense point cloud of toys images. Comparing it to the point cloud extracted from the original workflow (shown in Figure 5-17), the former (4,896,467 vertices) is much denser than the later (336,684 vertices), not to mention the former uses only two images, while the later uses 56 images. The second modification would increase the possibility of exacting keypoints for 3D registration because of much denser point cloud.



Figure 5-16 The dense point clouds of toys images from different view angles.



(a)

(b)

Figure 5-17 The comparison of the point clouds extracted by the original workflow and the workflow of the second modification. (a) is the point cloud from the second modification. (b) is the point cloud from the original workflow.

Applying the second modified workflow to Pictometry images, the dense point cloud computed from SGBM disparity map contains a lot of noise, shown as Figure 5-18. After adding a bilateral filter (window size: 30, $\sigma_d = 30$, and $\sigma_r = 1.5$) to the disparity map, the noise is highly reduced, but the buildings were twisted.

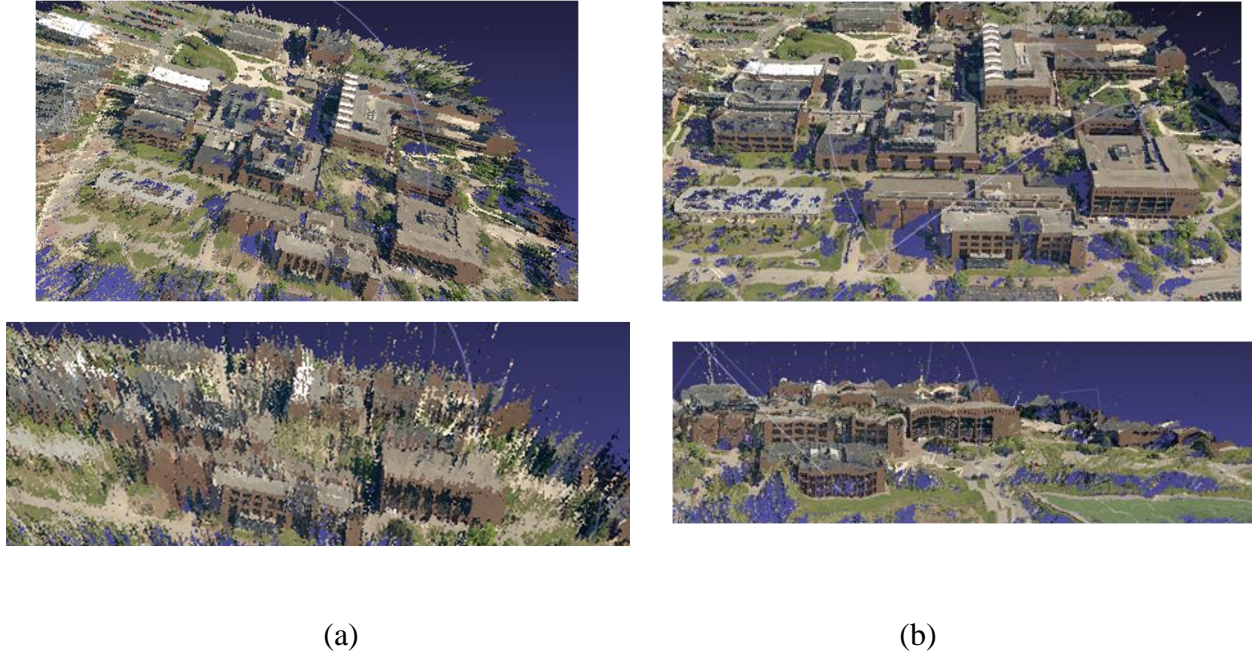


Figure 5-18 The dense point cloud of Pictometry images from different view angles. (a) is the point cloud before bilateral filtering. (b) is the point cloud after bilateral filtering.

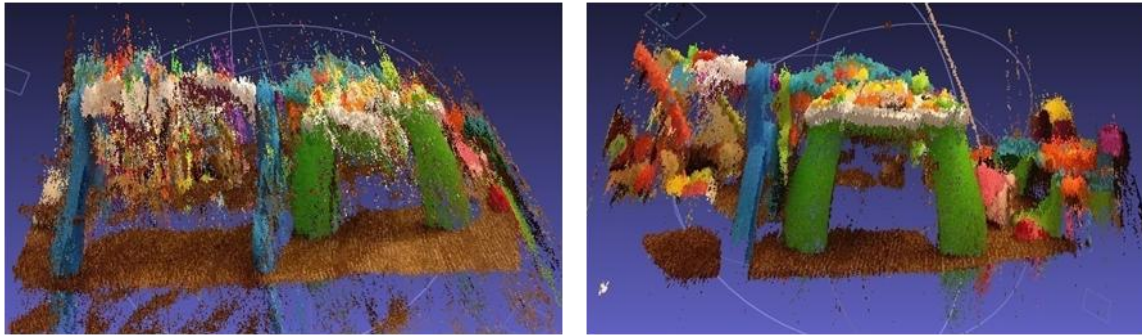
In order to find the reason of failure, three tests were done in this thesis.

1) The ratio of the baseline to the camera height

The base of this pair of Pictometry images is only 144 pixels, while the flight height is almost 1400m. The low ratio of the baseline to the camera height might be a problem, because errors would be brought in during the projection step if the ratio is low enough.

In order to check the effect of different disparity ranges on the accuracy of disparity maps, a pair of toys images with a very low baseline of 95 pixels is tested. The result shows in Figure 5-19. Comparing to the point cloud with the baseline of 186 pixels, it confirms that low ratio of the

baseline to the camera height will emphasize the error and noise. And apparently, the baseline should be not too large because enough overlapping area is need.



(a)

(b)

Figure 5-19 The dense point cloud extracted from toys images with different baseline. (a) is extracted from image pair with lower disparity range, (b) is extracted from image pair with higher disparity range.

2) *The ratio of the building height to the camera height*

The ratio of the building height to the camera height might be another reason, since it is much lower in the toys images. In order to discuss the effect of the ratio of building height to the flight height, some images of Mega and Lego blocks are tested and compared. In Figure 5-20, it is obvious that the dense point cloud extracted from Mega block images has lower noise than the dense point cloud extracted from Lego block images, from which may be concluded that a lower ratio of building height to the flight height would lead to more noise in the dense point cloud.

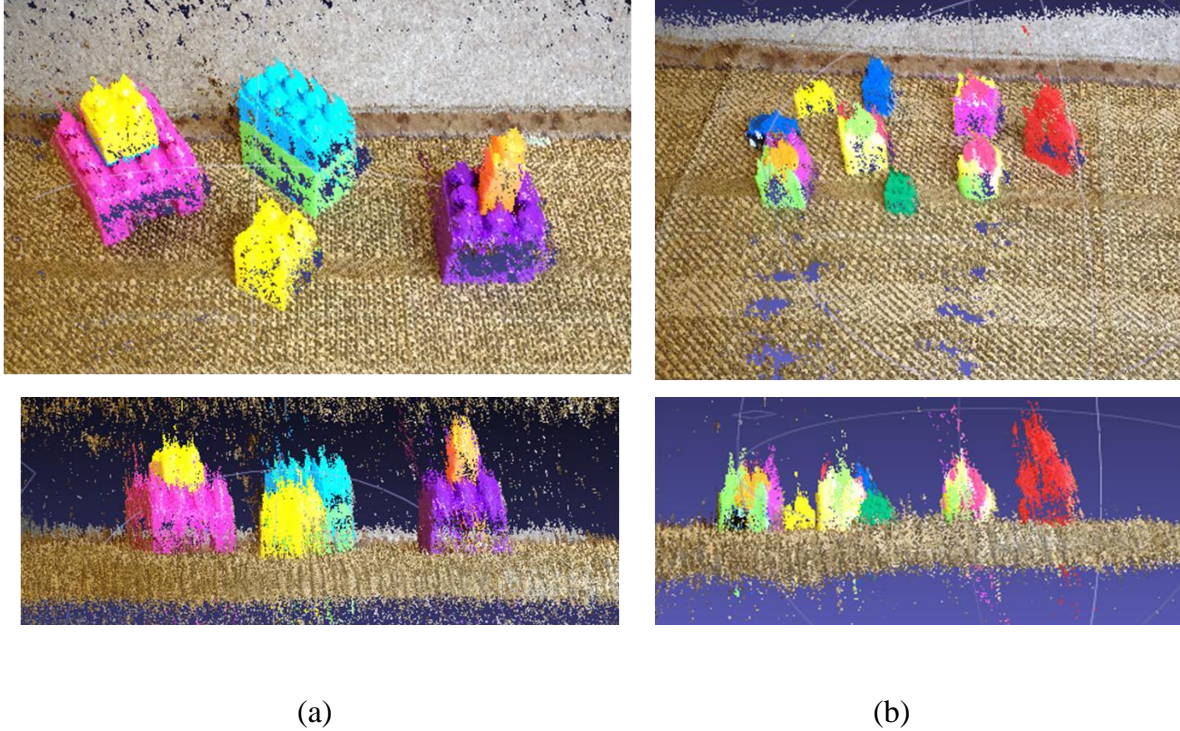


Figure 5-20 The dense point clouds. (a) is from Mega blocks, (b) is from Lego blocks.

3) *The approach for disparity map computation*

SGBM embedded in OpenCV is not the original SGM proposed by Hirschmuller (2008). There are some differences between these two approaches, as shown in Table 5-4. The cost calculation of original SGM is based on mutual information, while in OpenCV it is based on the Birchfield-Tomasi sub-pixel metric. The cost is aggregated from 8 or 16 directions in the original SGM, but only 5 or 8 directions in SGBM. The disparity maps of the test Tsukuba images computed by both SGM and SGBM are shown in Figure 5-21.

Table 5-4 The comparison of the original SGM and SGBM in OpenCV.

	Original SGM	SGBM in OpenCV
Color Image	No	Yes
Cost Function	Mutual information	Birchfield-Tomasi sub-pixel metric
Cost Aggregation	8 or 16 directions	5 or 8 directions
Matched Block Size	1	1 ... 11

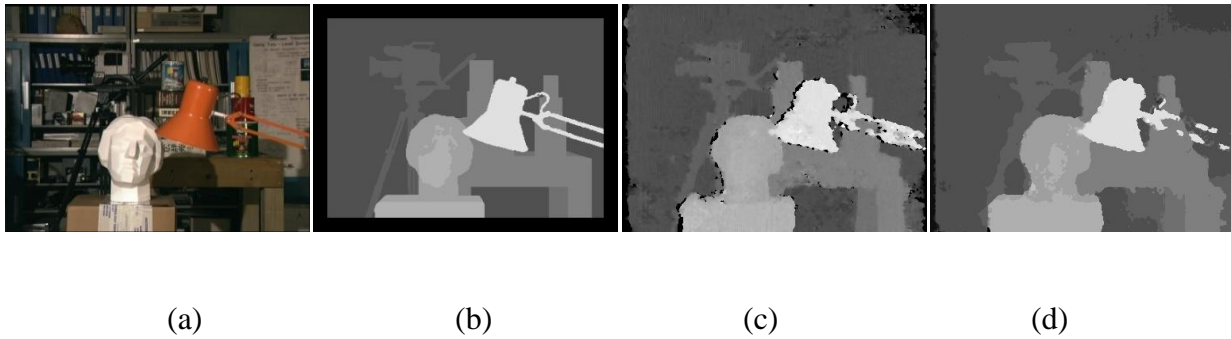


Figure 5-21 The disparity maps of Tsukuba images by both SGM and SGBM approaches. (a) is the image, (b) is the ground truth, (c) is the result from SGBM, (d) is the result from SGM.

Obviously, the result by SGM is better than the result by SGBM. The SGM result has higher accuracy, better edge and border response and more smoothness in the area with the same disparity. It is the same when applying the two approaches to the rectified toys images, shown in Figure 5-22.

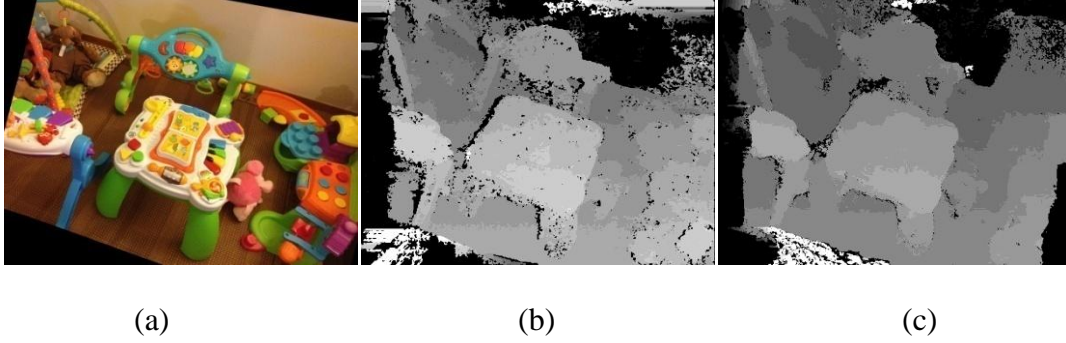


Figure 5-22 The disparity maps of toys images by both SGM and SGBM approaches. (a) is the rectified image, (b) is the result from SGBM, and (c) is the result from SGM.

In Pictometry oblique images, some factors cannot be changed, such as the building height or the flight height. The things we can do are finding image pairs with large baseline and using original SGM proposed by Hirschmuller. Another image pair with baseline of 1504 pixels was tried.

Since the original SGM is a memory consumption algorithm, Pictometry images are divided into several tiles for disparity computation, as Section 4.3.3.6 described. And all disparity sub-maps are merged together before projection. In this thesis, the overlap is 200 pixels, and tile size is 1000 rows by all the cols. Figure 5-23 shows the results of one tile. The SGM result is better than SGBM result with more valid pixels, and hence it has better border response.

The point clouds of this image pair is compared in Figure 5-24. Obviously, the point cloud extracted from SGM algorithm is denser than the one from SGBM class in OpenCV, and also it contains more wall points in the viewable direction.

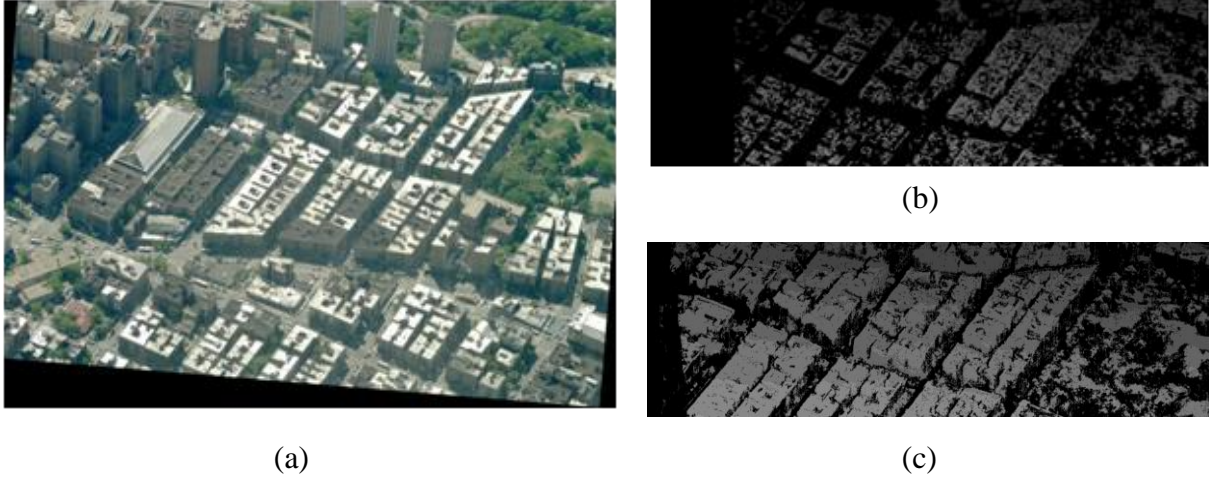


Figure 5-23 The disparity maps of Pictometry images by both SGM and SGBM approaches. (a) is the images after rectification, (b) is the results from SGBM, and (c) is the results from SGM.

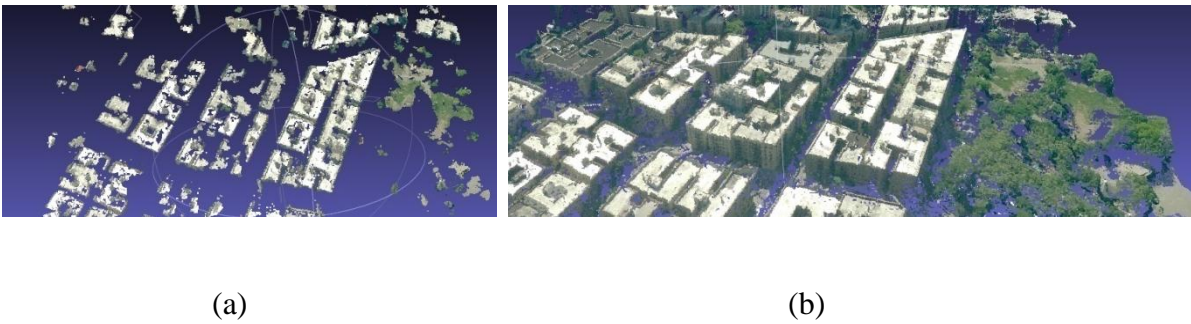
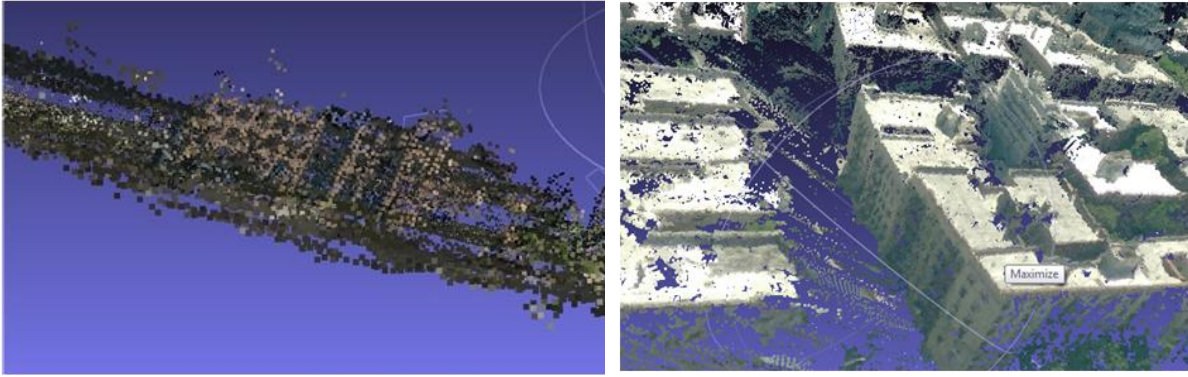


Figure 5-24 The point cloud projected from the disparity map computed by SGM. (a) is the result by SGBM, (b) is the result by SGM.

5.4 Results of the Third Modification - Noise Removal

In the point cloud from the first modification, Figure 5-25(a), many extraneous points are floating besides the walls and roofs. And in the dense point cloud from SGM, invalid points exist

through the ray of light (Figure 5-25(b)). These noise points would bring confusion in further processing steps such as surface extraction, roof measurement, and 3D feature detection. So noise removal step is needed.



(a)

(b)

Figure 5-25 Noise in the point cloud. (a) shows the extraneous points floating besides the walls and roofs, (b) shows invalid points exist as rays of light.

Two noise removal methods: the statistical noise removal method and the radius noise removal method are tested and compared in this thesis. Figure 5-26 gives the results. After noise removal, walls and roofs are much clearer in both results. For comparison, the same situations are set, such as the same number of neighbors and almost the same vertices after noise removal by adjusting the second parameters. No significant difference is detected in details in the point cloud of North-viewing images, as Figure 5-27 shows.



(a)

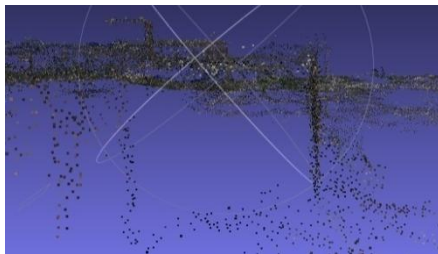


(b)

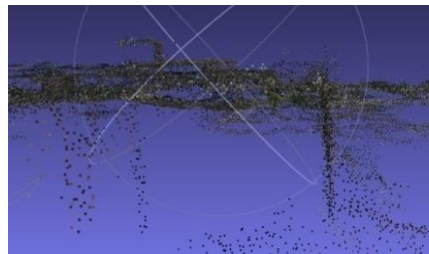


(c)

Figure 5-26 The results of noise removal. (a) is the point cloud with noise, (b) is the result of statistical removal method, (c) is the result of radius outlier removal method.



(a)



(b)

Figure 5-27 The comparison of the details of the two noise removal methods. (a) is the detail of the statistical removal method, (b) is the detail of the radius outlier removal method.

However, the point clouds of west and vertical images are much sparser than the North-viewing images because they were generated by just a few images. More vertices are removed in sparse areas by the radius outlier removal method than by the statistical removal method, even if they are correct points. The significant areas are marked by orange ellipses in Figure 5-28. So, the statistical noise removal method is better than the radius outlier noise removal method.



(a)



(b)



(c)



(d)

Figure 5-28 The noise removal results in sparse areas. (a) and (b) are the results of statistical removal method and radius outlier removal, respectively, in West-viewing point cloud, (c) and (d) are the results in vertical point cloud.

Applying statistical noise removal method to the dense point cloud from SGM, the result is shown in Figure 5-29. After noise removal, the noise on the light rays is almost removed, both between the buildings and down the earth.

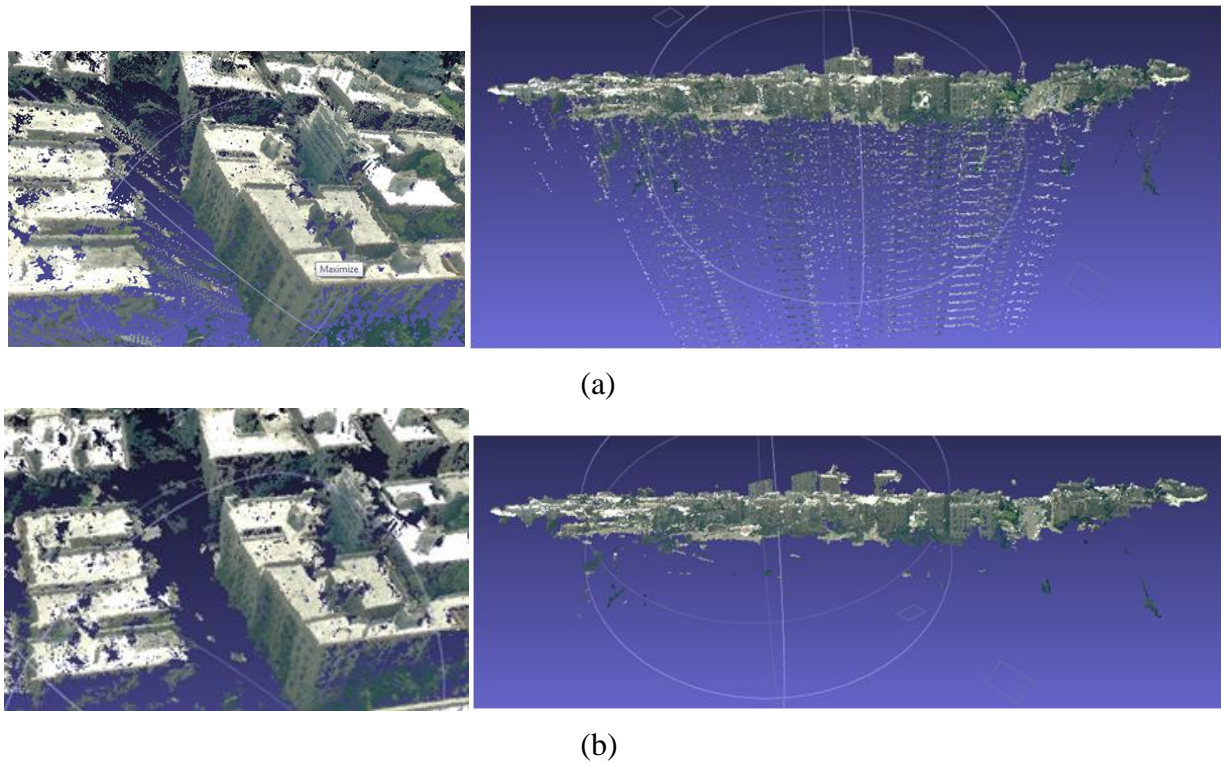


Figure 5-29 Dense point cloud after noise removal by statistical removal method. (a) is the point cloud with noise, (b) is the point cloud after noise removing by statistical removal method.

5.5 Accuracy Assessment

An ideal way for accuracy assessment is comparing the imagery-derived point cloud to LiDAR data, if we assume LiDAR data is the ground truth. However, 3D point cloud registration is

another challenge in the field, and also, most of the point clouds extracted in this thesis do not have corresponding LiDAR data. So, as a simple way, only the points from flat surfaces are selected for accuracy assessment, such as the subsets of points from flat roof, or flat ground. The accuracy is defined as the flatness of the points from a plane. For a better comparison of the accuracy of point clouds from the old workflow to the modified workflow, the point clouds from City Hall are selected, since they have dense point clouds from both workflows.

After selecting the subset of points within a flat surface, a plane is fitted to the points by RANSAC. The error is quantified as Mean Squared Error (MSE):

$$MSE = \frac{\sum_{i=1}^N \|p_i - \hat{p}_i\|^2}{N} \quad (5-1)$$

Here, p_i is the coordinate of point in the subset, \hat{p}_i is the projected coordinate of point p_i to the fitted plane, $\|\cdot\|^2$ is the Euclidian distance of the two points, and N is the total points in the subset.

The selected corresponding planes from the City Hall point clouds are shown as Figure 5-30. For the up plane, 239 points are selected from the point cloud extracted by the original workflow, while 5,579 points are selected from the point cloud by the modified workflow. And for the down plan, 373 points are selected from the point cloud extracted by the original workflow, while 10,953 points are selected from the point cloud by the modified workflow.

After fitting by RANSAC with the same parameters, the inliers and outliers for each subset points are shown in Figure 5-31. The units of the point coordinates are unknown, since the camera parameters are estimated by Bundler.

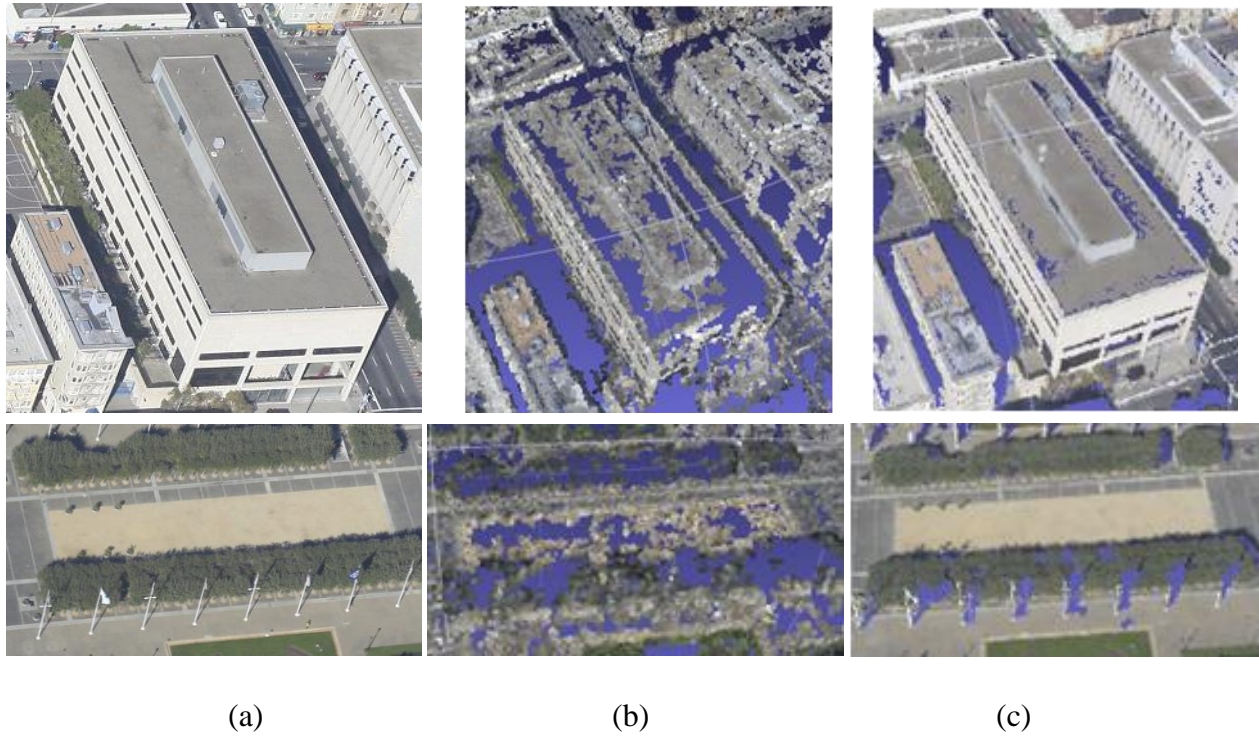
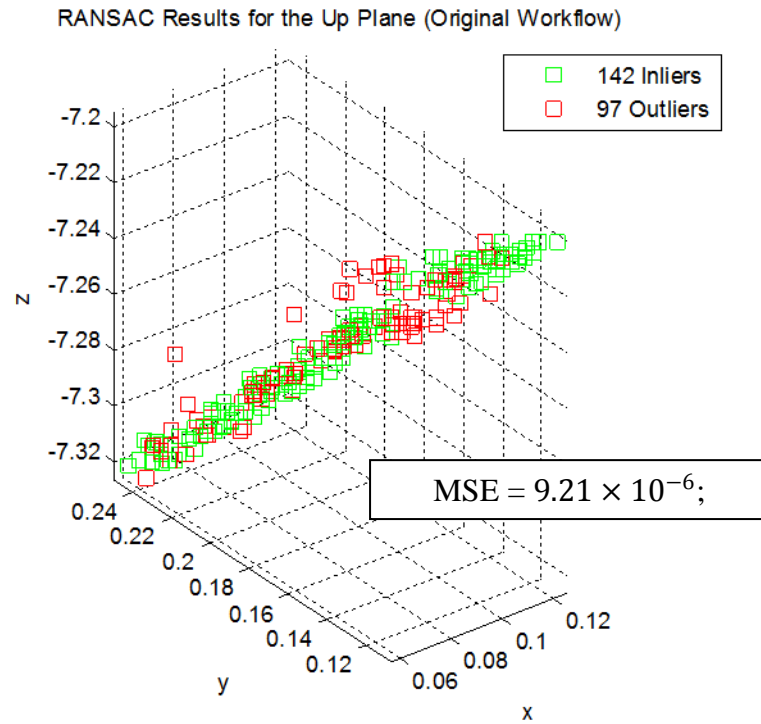


Figure 5-30 The point subsets from the point clouds extracted by the original workflow and the modified workflow. (a) are the corresponding image parts, (b) are the point subsets from the point clouds extracted by the original workflow, (b) are the point subsets from the point clouds extracted by the modified workflow.

(1a)



(1b)

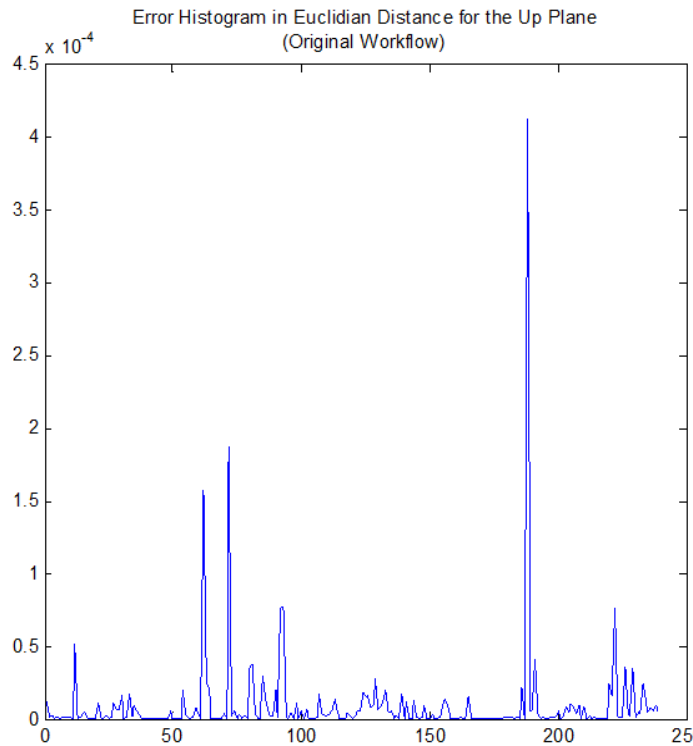


Figure 5-31 The inliers and outliers for each subset points after fitting by RANSAC with the same parameters. (1a) is the RANSAC results for the up plane (original workflow). (1b) is the error histogram in Euclidian distance for the up plane (original workflow).

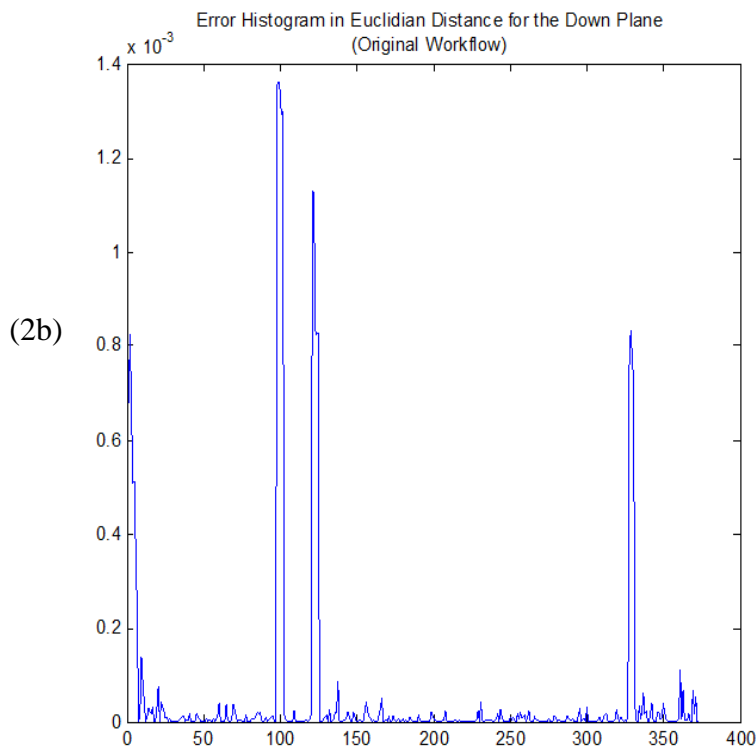
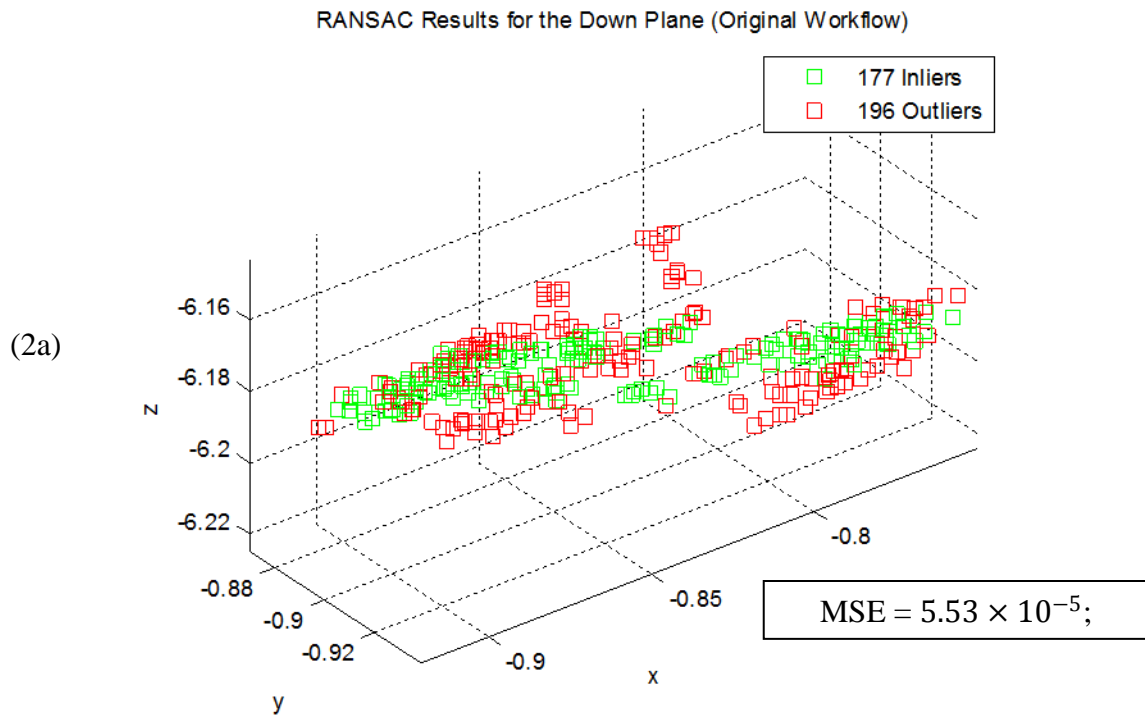
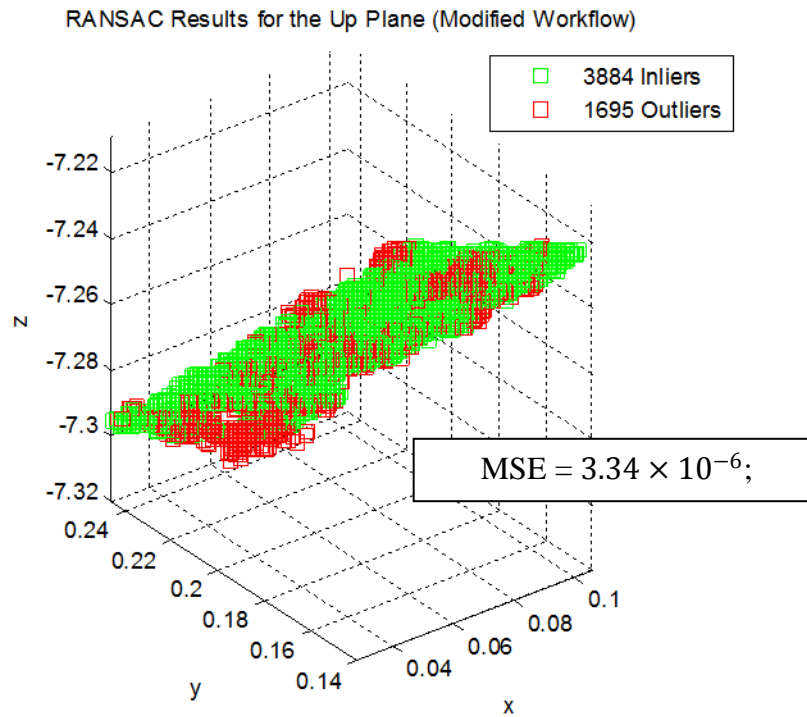


Figure 5-31 (Cont'd) The inliers and outliers for each subset points after fitting by RANSAC with the same parameters. (2a) is the RANSAC results for the down plane (original workflow). (2b) is the error histogram in Euclidian distance for the down plane (original workflow).

(3a)



(3b)

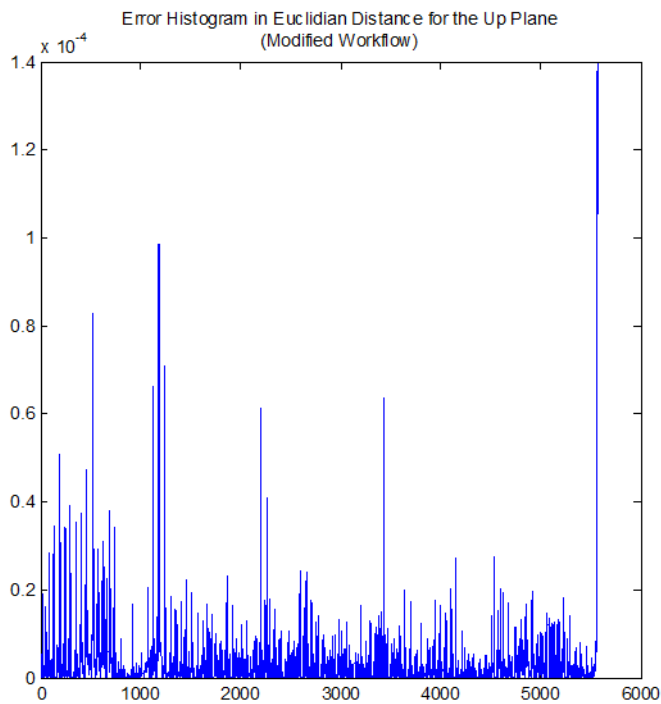
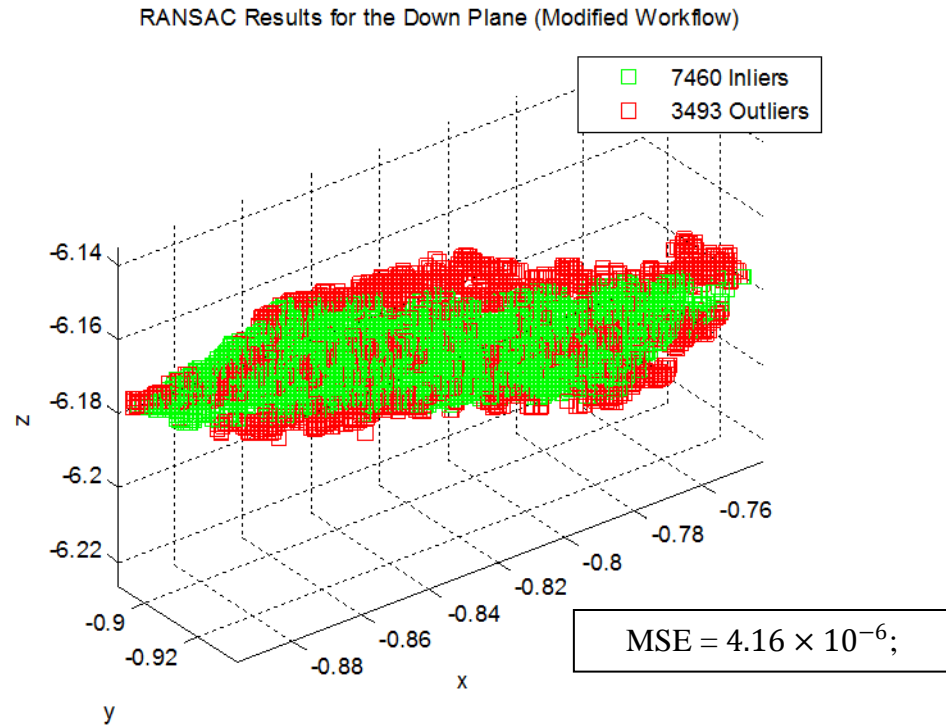


Figure 5-31 (Cont'd) The inliers and outliers for each subset points after fitting by RANSAC with the same parameters. (3a) is the RANSAC results for the up plane (modified workflow). (3b) is the error histogram in Euclidian distance for the up plane (modified workflow).

(4a)



(4b)

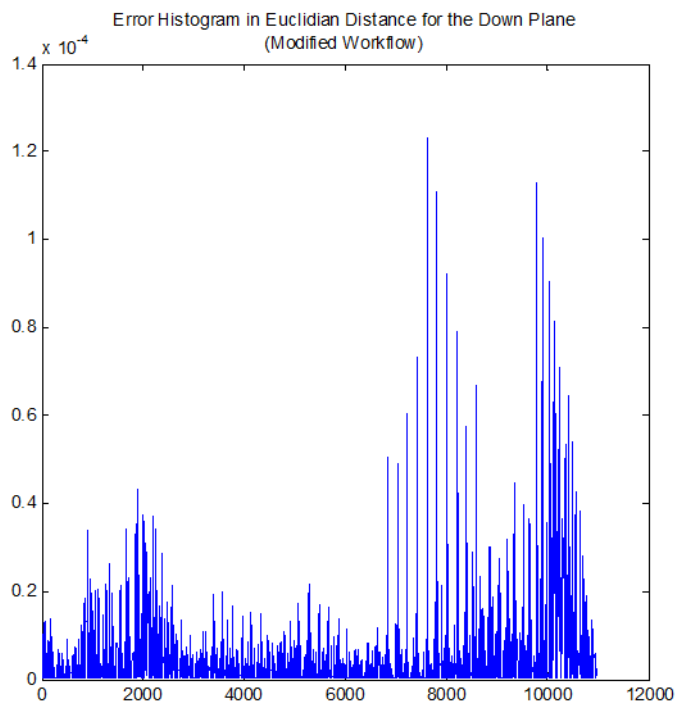


Figure 5-31 (Cont'd) The inliers and outliers for each subset points after fitting by RANSAC with the same parameters. (4a) is the RANSAC results for the down plane (modified workflow). (4b) is the error histogram in Euclidian distance for the down plane (modified workflow).

In the figures, the green points are inliers while the red points are outliers. We can see, the point cloud subsets from the modified workflow are much denser than the point cloud subsets from the original workflow. And the inliers ratios of the point subsets from the modified workflow, 0.70 and 0.68 ($3884/(3884+1695)$ and $7460/(7430+3493)$), are also higher than the ratios of point subsets from the original workflow, 0.59 and 0.47 ($142/(142+97)$ and $177/(177+196)$), respectively. Most importantly, the MSEs of the point subsets from the modified workflow, 3.34×10^{-6} and 4.16×10^{-6} , are slightly lower than the MSEs of the point subsets from the original workflow, 9.21×10^{-6} and 5.53×10^{-5} , respectively. With the error histograms, we can conclude that the point subsets from the modified workflow are more flat than the ones from the original workflow.

6 Conclusion

Oblique images taken from different view directions contain more building facade information than nadir images. The availability of oblique aerial images brings the possibility to extract dense point clouds with both roof points and wall points automatically. A more dense point cloud is desired for 3D point cloud registration, surface detection, and block model creation.

In this thesis, based on an earlier developed RIT workflow, three modifications are proposed and tested for extracting a dense point cloud from oblique images automatically. The first modification is to replace the first part of the workflow from SIFT to ASIFT. The result reveals that this modified workflow extracts 40% more vertices than the RIT workflow from Pictometry oblique images. Then, floating walls are corrected by fixing the focal lengths for each image individually.

The second modification is implementing SGM to do dense stereo matching and disparity map computation. Then, we project the disparity map back to 3D point cloud by using the basic photogrammetry model and triangular similarity. Initially, the SGBM function imbedded in OpenCV was used to compute the disparity map. The comparison of the point cloud extracted by SGM to the point cloud by the original workflow shows that SGM could give a much more dense point cloud. However, since the dense point cloud extracted from Pictometry oblique images contains too much noise to remove even by a bilateral filter on the disparity map, three tests were made. The testing results show that, firstly, the ratio of the baseline to the camera height should be in a reasonable range, since that too small of a ratio would enlarge the errors in

projection and too large a ratio would reduce the overlapping area. Secondly, a low ratio of the building height to the camera height would also bring noise to the dense point cloud. Also, the approach for disparity map computation should be the original SGM proposed by Hirschmuller (2008), not the SGBM in OpenCV, since SGM gives a better result with higher accuracy, better edge and border response and more smoothness in the areas with the same disparity. After these discussions, another pair of Pictometry oblique images was selected, and the point cloud as dense as expected with clear border response.

The third modification is noise removal. Two methods are tested and compared. The result shows that the two methods are the same in dense point areas, while the statistical removal method has better results in the sparse areas.

To evaluate the modified workflow, an accuracy assessment is made in the end. From the point clouds extracted by the original workflow and the modified workflow, respectively, point subsets from two flat areas are selected. By fitting planes with RANSAC, the Mean Squared Errors (MSEs) of the points to the fitted planes are compared, respectively. The point subsets from the modified workflow have higher percentages of inliers and lower MSEs than the ones from the original workflow, respectively.

7 Future Work

In this thesis, a successful attempt is made to extract dense point clouds from oblique images automatically. Starting from the previous work done by RIT researchers, three modifications were proposed and tested. The results demonstrated in Section 5 shows that the dense point cloud extracted by the modified workflow is much denser than the one from the original workflow, and it has slightly higher accuracy on flat surfaces.

Since the point cloud extraction failed for some Pictometry image pairs, three possible reasons were discussed. The discussion shows that the SGM algorithm is sensitive to the ratio of the baseline to the camera height and the ratio of the building height to the camera height. However, in this thesis, we did not determine what ranges of these two ratios are best. So in the future, research should be pursued on how sensitive these two ratios would be and their best ranges in processing oblique images, especially airborne oblique images such as Pictometry oblique images.

From in Table 5-2 in Section 5.2.2 we can see that only one camera's focal length was not fixed when `constrain_focal_weight` was set to `1.0e6`. It might indicate that the keypoints computed from that unfixed image are not good enough for adjusting in bundler. That could be used to eliminate bad images.

In the future, a better way should be used to assess the accuracy of the dense point cloud, such as comparing it to a corresponded LIDAR data. For experimental assessment, the Digital Imaging

and Remote Sensing Image Generation (DIRSIG) Model (DIRSIG, 2013), developed by the Digital Imaging and Remote Sensing Laboratory at Rochester Institute of Technology, is a perfect model for practicing and testing. It produces passive single-band, multi-spectral or hyper-spectral imagery from the visible through the thermal infrared region of the electromagnetic spectrum with camera information, and it also has a very mature active laser (LIDAR) capability and an evolving active RF (RADAR) capability. By using the data provided by the DIRSIG Model, we can check the accuracy of estimated camera parameters and the accuracy of the dense point cloud.

Also, since 3D modeling is more and more popular nowadays, in cameras (3D camera) or in cell phones in the future, the time and memory required should be within a reasonable range, or even better, realizable with real time processing. However, the SGM is a time and memory consuming algorithm. It would be better if a faster and more memory efficient version could be implemented in the future.

References

- Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., and Szeliski, R. 2009. Building rome in a day, *Computer Vision, 2009 IEEE 12th International Conference*, pp. 72-79.
- Amo, M., Martinez, F., and Torre, M. 2006. Road extraction from aerial images using a region competition algorithm, *Image Processing, IEEE*, 15(5):1192-1201.
- Brown, M. and Lowe, D.G. 2002. Invariant features from interest point groups. *In British Machine Vision Conference*, Cardiff, Wales, pp. 656-665.
- Cheng, L., Gong, J., Li, M., and Liu, Y. 2011. 3D building model reconstruction from multi-view aerial imagery and LiDAR data. *Photogrammetric Engineering & Remote Sensing*, 77(2):125-139.
- DIRSIG. 2013. Available at <http://dirsig.org/>.
- Fischler, M.A., and Bolles, R.C. 1981. RANSAC random sampling consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 26:381-395.
- Gerke, M. 2009. Dense matching in high resolution oblique airborne images, *CMRT09: Object Extraction for 3D City Models, Road Databases and Traffic Monitoring - Concepts, Algorithms, and Evaluation*, IAPRS, Paris, France. Vol. XXXVIII, Part 3/W4

- Gerke, M. and Kerle, N. 2011. Automatic structural seismic damage assessment with airborne oblique pictometry imagery. *Photogrammetric Engineering & Remote Sensing*, 77(9):885-898.
- Gerke, M. and Nyaruhuma, A. 2009. Incorporating scene constraints into the triangulation of airborne oblique images. *In: High-Resolution Earth Imaging for Geospatial Information, Vol. XXXVIII, International Archives of Photogrammetry and Remote Sensing, Hannover.*
- Gill, C. 2010. Aerial perspective: surveying and modeling hard-to-measure sites. *Professional Surveyor Magazine.*
- Haala, N. and Kada, M. 2010. An update on automatic 3D buildingreconstruction, *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580.
- Habbecke, M. and Kobbelt, L. 2010. Automatic registration of oblique aerial images with cadastral maps. *ECCV Workshop on Reconstruction and Modeling of Large Scale 3D Virtual Environments.*
- Hartley, R., and Zisserman, A. 2004. *Multiple view geometry in computer vision (2nd ed.)*. Cambridge: Cambridge University Press ISBN:0521540518.
- Heinrichs, M., Hellwich, O. and Rodehorst, V. 2007. Efficient semi-global matching for trinocular stereo. *In: IAPRS*, 36:185-190.

- Hirschmuller, H. 2003. Stereo Vision Based Mapping and Immediate Virtual Walkthroughs. *School of Computing De Montfort University Leicester, LE1 9BH, UK*. 34-35.
- Hirschmuller, H. 2008. Stereo processing by semi-global matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328-341.
- Hover Inc, 2013. www.hoverinc.com/.
- Jurisch, A. and Mountai, D. 2008. Evaluating the viability of Pictometry imagery for creating models of the built environment. *ICCSA '08 Proceeding of the international conference on Computational Science and Its Applications, Part I*, pp. 663-677.
- Kim, J., Kolmogorov, V., and Zabih, R. 2003. Visual correspondence using energy minimization and mutual information. *IEEE International Conference on Computer Vision*.
- Kolaric, S. 2007. Camera's coordinate system in OpenCV: <http://cv-kolaric.blogspot.com/2007/06/cameras-coordinate-system.html>
- Lindeberg, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224-270.
- Lourakis, M. and Argyros A. 2004. *The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm*. Institute of Computer Science, 4th edition.

- Lowe, D.G. 1999. Object recognition from local scale-invariant features. *In International Conference on Computer Vision*, Corfu, Greece, pp. 1150-1157.
- Lowe, D.G. 2004. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, 60(2):91-110.
- Luong, Q., and Faugeras, O.D. 1996. The fundamental matrix: theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17:43-75.
- Morel J.M. and Yu, G. 2009. ASIFT, A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438-469.
- Nilosek, D.R. and Salvaggio, C. 2010. Applying computer vision techniques to perform semi-automated analytical photogrammetry. *Image Processing Workshop (WNYIPW)*, Western New York, pp. 1-5.
- OpenCV. 2013. Available at <http://opencv.willowgarage.com/wiki/>.
- PCL. 2013. Available at <http://pointclouds.org/>.
- Remondino, F. and El-Hakim, S. 2006. Image-based 3D modeling a review, *The Photogrammetric Record*, 21(115):269-291.
- Scharstein, D. and Szeliski, R. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7-42

- Secord, J. and Zakhor, A. 2007. Tree detection in urban regions using aerial LiDAR and image data. *Geoscience and Remote Sensing Letters, IEEE*, 4(2):196-200.
- Sirmacek, B. and Unsalan, C. 2008. Building detection from aerial images using invariant color features and shadow information. *Computer and Information Sciences, 2008. ISCIS '08. 23rd*, pp. 1- 5.
- Smith, M.J., Hamruni, A.M., and Jamieson, A. 2009. 3-D urban modeling using airborne oblique and vertical imagery. *ISPRS Hannover Workshop 2009. High-Resolution Earth Imaging for Geospatial Information*, Hannover, Germany, Vol. XXXVIII-1-4-7/W5.
- Snavely, N., Seitz, S.M., and Szeliski, R. 2006. Photo tourism: exploring image collections in 3D. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006)*.
- Triggs, B., McLauchlan, P.F., Hartley, R.I., and Fitzgibbon, A.W. 2000. Bundle adjustment — a modern synthesis. *Vision Algorithms: Theory AND Practice, Lecture Notes in Computer Science*, (1883/2000):153-177.
- Tsai, V.J.D. 2006. A comparative study on shadow compensation of color aerial images in invariant color models. *Geoscience and Remote Sensing, IEEE*, 44(6):1661-1671.
- Walli, K.C., Nilosek, D.R., Schott, J.R., and Salvaggio, C. 2009. Airborne synthetic scene generation (AeroSynth). *Proceedings of the ASPRS, ASPRS/MAPPS 2009 Fall Conference, Digital Mapping - From Elevation to Information, Digital Elevation Data Fusion Innovations, San Antonio, Texas, United States*.

- Wang, Y., Schultz, S., and Giuffrida, F. 2008. Pictometry's proprietary airborne digital imaging system and its application in 3D city modeling. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XXXVII. Part B1.
- WASP. 2013. Available at <http://lias.cis.rit.edu/projects/wasp>.
- Wu, C., Agarwal, S., Curless, B., and Seitz, S.M. 2012. Schematic surface reconstruction. *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE*, pp. 1498-1505.
- Furukawa, Y. and Ponce, J. 2009. Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. VOL. 32, NO. 8, pp. 1362-1376.
- Zhang, Y., Li, J., and Song, P. 2011. 3D Road information extraction from LiDAR data fused with aerial-images. *Spatial Data Mining and Geographical Knowledge Services (ICSDM), 2011 IEEE International Conference*, pp. 362-366.
- Zou, L. and Li, Y. 2010. A method of stereo vision matching based on OpenCV. *Audio Language and Image Processing (ICALIP), 2010 International Conference*, pp. 185-190.