3D CITY SCALE RECONSTRUCTION USING WIDE AREA MOTION IMAGERY

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

RAPHAEL VIGUIER

Dr. Kannappan Palaniappan, Thesis Supervisor

JULY 2018

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

3D CITY SCALE RECONSTRUCTION USING WIDE AREA MOTION IMAGERY

presented by Raphael Viguier,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Kannappan Palaniappan

Dr. Jeffrey Uhlmann

Dr. Filiz Bunyak Ersoy

Dr. David R. Larsen

To my parents and my brothers, none of this would have been possible without your unconditional love and support, from my first day on earth to these last years spent so far away from you.

To Heather, I am so thankful for having you by my side during the past 5 years and I could not imagine going through these without you.

To Henry, my dedicated "pair programming" partner.

ACKNOWLEDGMENTS

I want to express my gratitude to my advisor Dr. Kannappan Palaniappan for his advice, and support during all these years first as an intern and then as a graduate student. I also owe to him the opportunity I had to work as a research intern at IBM. I learnt a lot as his students and feel particularly well prepared for my next endeavors thanks to him. I also want to thank Dr. Guna Seetharaman, and Dr. Steve Suddarth who made it possible for me to work on such an interesting research topic.

I would like to thank Dr. Uhlmann and Dr. Larsen for doing me the honor of being members of my doctoral committee and for the invaluable insights they provided to improve this research work. I want to thank Dr Filiz Bunyak for being a member of my committee too, but also for all the particular help and support she provided while I worked with her at the CIVA lab.

I would like to thank everyone at the CIVA lab: Rengarajan, Hadi, Surya, Mahdieh, Sema, Anoop, Rahul, Josh, Yao, Ke, Rumana, and Rui. It was great working with all of you and you made this journey so much nicer.

I would also like to thank all my collaborators at IBM, in particular Sharathandra Pankanti, Chung Ching Lin and Karthikeyan Ramamurthy. Working at IBM was an invaluable experience, during which I learned a lot from all of you.

I also want to thank My CEO and CTO, Tara Pham and Ilan Goodman as well as all the Numina team for their support towards finishing my degree.

Finally , I also want to particularly thank Jodie Lenser for her advice and the countless times she helped me during all these years as a graduate student.

TABLE OF CONTENTS

ACKNOWLEDGMENTS . . . . . **ii**

LIST OF TABLES . . . . . **viii**

LIST OF FIGURES . . . . . **ix**

ABSTRACT . . . . . **xiii**

CHAPTER . . . . .

1 Introduction **1**

1.1 3D Reconstruction in Wide Area Motion Imagery 5

1.1.1 From 2D Registration to 3D Reconstruction 6

1.1.2 Challenges 6

1.1.3 Contribution 10

1.2 Datasets Overview 12

1.2.1 Four Hills 13

1.2.2 Albuquerque 16

1.2.3 Berkeley 18

1.2.4 Los Angeles 20

1.2.5 Columbia 22

1.3 Dataset Preprocessing 24

1.3.1 Using Sensor Metadata 24

1.4 Thesis Outline 30
1.5 Notations 31
2 Fundamentals of 3D Computer Vision 32
2.1 Pinhole Camera Model 32
2.1.1 The Intrinsic Matrix **K** 33
2.1.2 The Rotation Matrix **R** 36
2.1.3 The Translation Vector **t** 37
2.1.4 Projection Matrix 37
2.1.5 Bundle Adjustment Formulation 39
2.2 Camera Calibration 39
2.3 Epipolar Geometry and Image Transformation 40
2.3.1 Homography 40
2.3.2 Epipolar Geometry 41
2.3.3 Fundamental and Essential Matrices 42
2.3.4 4 Points and 8 Points Algorithms 43
2.4 Data Representation 43
2.4.1 Depthmap 43
2.4.2 Voxel Grid 44
2.4.3 Point Cloud and Mesh 46
2.5 Data Visualization 48
3 Related Work 51
3.1 Bundle Adjustment 51
3.2 Stereo 3D Reconstruction 52
3.3 Patch Based Methods 53
3.4 Volumetric Methods 54

3.5 Surface Fitting Method 56
3.6 Other Notable Work 56
3.6.1 Single View 3D Reconstruction 56
3.6.2 Deep Neural Network 57
3.7 Site Modeling 58
3.8 Thesis Contribution 58
4 Voting Pipeline 60
4.1 Introduction 60
4.2 Pipeline Description 61
4.2.1 Voting 63
4.3 Post-Processing 65
4.3.1 Vote Collapsing 65
4.3.2 Gradient Magnitude 66
4.3.3 Horizontal Smoothing 67
4.3.4 Histograms Computing 67
4.3.5 Combination of Information for Automatic Thresholding . . . 67
4.3.6 Manual Thresholding 70
4.4 Point Cloud Coloring 71
4.4.1 Bresenham Based Coloring Algorithm 72
4.4.2 Z-Buffer Based Coloring Algorithm 72
4.4.3 Complexity Analysis 74
4.5 Experimental Results 75
4.5.1 Four Hills 75
4.5.2 Albuquerque 75
4.5.3 Berkeley 76

4.5.4 Los Angeles 77
4.5.5 Columbia 77
4.5.6 Scaling Summary 78
4.6 Other Experiments 79
4.6.1 Temporal Incremental Voting and Removal 79
4.6.2 Occlusion and Appearance Consistency Filtering 82
4.6.3 Thresholding by Vote Accumulation 84
4.6.4 Algorithms Memory Requirement 85
4.7 Discussion: Shortcoming and Limitations 86
5 Plane Sweep with Voting Pipeline 88
5.1 Introduction 88
5.2 Pipeline Overview 88
5.2.1 Plane Sweeping 90
5.2.2 Depth Based Voting 94
5.2.3 Algorithms Memory Requirement 94
5.2.4 Point Cloud Extraction 95
5.2.5 Depth Voting and Thresholding 95
6 Benchmarking and Evaluation 101
6.1 Evaluation of Accuracy and Completeness of the 3D Structure 102
6.2 Evaluation of the Photo Consistency 102
6.3 Qualitative Comparison 103
6.3.1 Evaluation using cloud compare 103
7 Applications 106
7.1 Introduction 106
7.2 Altitude Masks 106

7.2.1 Synthetic Images 109
7.2.2 Hazard Map 109
7.3 Ortho-rectified Maps 110
8 Conclusion 113
BIBLIOGRAPHY 116
VITA 127

LIST OF TABLES

1.1 Four Hills summary 14
1.2 Albuquerque summary 16
1.3 Berkeley summary 18
1.4 Los Angeles summary 20
1.5 Columbia summary 22

4.1 Summary of Dataset resolution and output point cloud sizes 78

LIST OF FIGURES

1.1 General principle of 3D Reconstruction 1

1.2 Example of 3D reconstruction in medical imaging 2

1.3 Example of WAMI data 7

1.4 An illustraton of parallax in registered images 8

1.5 Illustration of the combination of camera pose error estimate 9

1.6 An example of 3D optical illusion 10

1.7 An example illustrating scene uncertainty 11

1.8 An overview of MU3D pipeline 12

1.9 Illustration of WAMI data acquistion conditions 13

1.10 Example of frames from Four Hills 15

1.11 Example of frames from Albuquerque 17

1.12 Example of frames from Berkeley 19

1.13 Example of frames from Los Angeles 21

1.14 Example of frames from Columbia 23

1.15 Lamppost manually tracked in Four Hills dataset 25

1.16 Ground projection calculation 25

1.17 Ground Projection Illustration 26

1.18 Visualization of epipolar lines on Four Hills 26

1.19 Top view projection of lamppost rays 27

1.20 Generation of a synthetic path 28

1.21 Synthetic data experiment results 29

2.1 The Pinhole camera model 33
2.2 Camera aspect ratio 34
2.3 Example of camera distorsions 36
2.4 World and camera system of coordinates 38
2.5 A checkboard used for camera calibration 40
2.6 Illustration of stiching two images using Homography 41
2.7 Epipolar geometry 42
2.8 An illustration of depthmaps 44
2.9 Viisualization of a voxel grid on a real dataset 45
2.10 Illustration of rasterization 46
2.11 Illustration of octree 46
2.12 Illustration of Delauney triangulation in Meshlab 47
2.13 A visualization the Hough transform space for a line 49
2.14 Standford bunny visualized in meshlab 50

3.1 Distributed bundle adjustment illustration 52
3.2 Local plane sweep result 54
3.3 PMVS result 55
3.4 Crispell et Al novel view generation result 56
3.5 SSD applied to MU3D Los Angeles point cloud 57
3.6 An example of output from PlaneNet 58
3.7 Screenshot of Empire State Building and its surroundings in Google map 59

4.1 Block diagram of the voting pipeline 62
4.2 Block diagram focusing on dense reconstruction 62
4.3 Illustration of the voting principle 63
4.4 Example of rays not well separated 64

4.5 Example of well separated rays 65
4.6 Comparison with and without vote collapsing 66
4.7 Histograms of votes and gradient magnitude for Albuquerque 68
4.8 Example of sigmoid 69
4.9 Example of top views used for thresholding 71
4.10 Relationship between depth gradient and surface normal 73
4.11 Screenshots of the vizualized output for Four Hills 75
4.12 Screenshots of the vizualized output for Albuquerque 76
4.13 Screenshots of the vizualized output for Berkeley 76
4.14 Screenshots of the vizualized output for Los Angeles 77
4.15 Screenshots of the vizualized output for Columbia 78
4.16 Example of incremental voting and removal 81
4.17 Probability function along a ray 83
4.18 Occlusion and Apperance consistency filtering result 84
4.19 Example of result for vote accumulation thresholding 85

5.1 Block Diagram of Plane Sweeping Pipeline 89
5.2 Plane sweeping principle 89
5.3 Illustration of plane sweeping implementation 90
5.4 Image warping transformation 91
5.5 Image scaling transformation 92
5.6 Plane sweep image difference 92
5.7 Example of output depthmaps from Albuquerque 93
5.8 Illustration of depthmap fusion 94
5.9 Albuquerque Vote and visibility count histograms 96
5.10 Top views of vote space and visibility count for Albuquerque. 97
5.11 Screenshot of final point cloud for Albuquerque 98
5.12 Screenshot of final point cloud for Columbia MO 99

5.13 Multiple iteration with prior 100

6.1 Learning based methods results 103

6.2 Point cloud evaluation in Cloud Compare 105

7.1 Superimposed altitude mask on its corresponding image for Albuquerque dataset. 107

7.2 Altitude masks illustration 108

7.3 Example of synthetic images generated with Los Angeles point cloud 110

7.4 Sampling of the superior hemisphere using recursive divisions 110

7.5 Hazard map for Los Angeles, hemisphere sampled with 105 points . . 111

7.6 Illustration of 3D orthorectified image for Albuquerque 112

ABSTRACT

3D reconstruction is one of the most challenging but also most necessary part of computer vision. It is generally applied everywhere, from remote sensing to medical imaging and multimedia. Wide Area Motion Imagery is a field that has gained traction over the recent years. It consists in using an airborne large field of view sensor to cover a typically over a square kilometer area for each captured image. This is particularly valuable data for analysis but the amount of information is overwhelming for any human analyst. Algorithms to efficiently and automatically extract information are therefore needed and 3D reconstruction plays a critical part in it, along with detection and tracking.

This dissertation work presents novel reconstruction algorithms to compute a 3D probabilistic space, a set of experiments to efficiently extract photo realistic 3D point clouds and a range of transformations for possible applications of the generated 3D data to filtering, data compression and mapping. The algorithms have been successfully tested on our own datasets provided by Transparent Sky and this thesis work also proposes methods to evaluate accuracy, completeness and photo-consistency. The generated data has been successfully used to improve detection and tracking performances, and allows data compression and extrapolation by generating synthetic images from new point of view, and data augmentation with the inferred occlusion areas.

Chapter 1

Introduction

Three-dimensional (3D) reconstruction is the art of estimating a model of the shape and the appearance of a real object through image processing. It is one of the prominent area of computer vision, finding many applications in various domains, such as remote sensing [1], medical imaging [2, 3] (figure 1.2), multimedia [4]. There are many different approaches depending on the problem that is being solved and the means at disposal but it is however always possible to distinguish two principal layers between

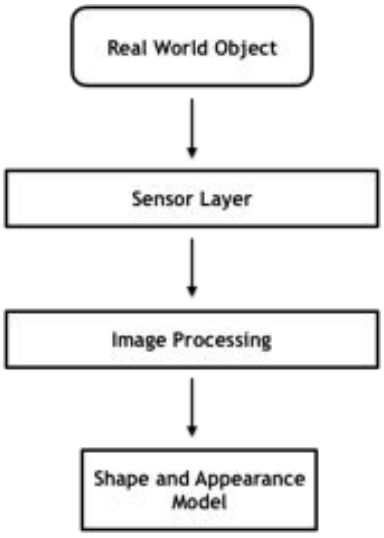


Figure 1.1: From physical world to digitalized 3D model.

the physical world and the data being produced: the sensor layer which determines what type of data is then used by the image processing layer (see figure 1.1).

Several critical tasks can be decided to be handled in either one of these two layers. For example, depth estimation can be obtained directly from a depth sensor at the hardware layer, or inferred from the images using image processing software. On the sensor layer, a distinction is often made between active methods that interact with the physical world by analyzing the reflection of a signal they directly emitted and passive methods whose sensor only measures the sun light reflected by the object. As the most prominent active method, LIDAR (Light Detection and Ranging, or contraction of Light and RADAR) consists in measuring distance of an object, by reflecting a laser on its surface, and measuring the difference in wavelengths and return time, in order to infer the distance. It is possible to achieve high resolution depth imaging with this method. LIDAR data when available are often a preferred way of acquisition that can be used to benchmark other reconstruction method.

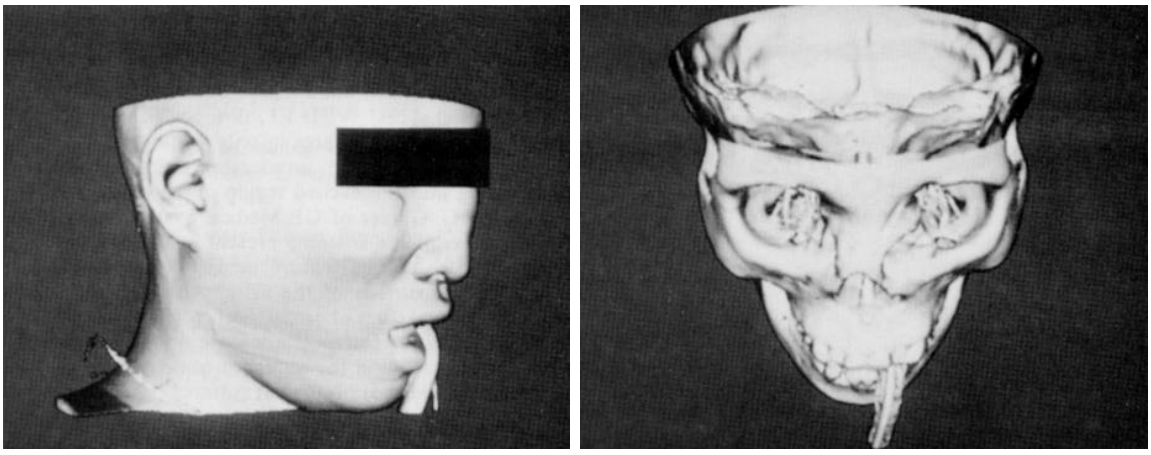


Figure 1.2: An example of result of Marching Cube applied to medical imaging [3]

Expectedly, the design of the software layer is heavily dependent on the upstream sensor layer. It can contain various subtasks depending again on the sensor layer but also on the final goal. It is however possible to distinguish three main categories that we will name as follows:

- camera pose estimate
- shape or structure modeling
- novel view generation

Depending on the application, one or several of these tasks are necessary. The need for camera pose estimation can be avoided if the data acquisition is done in a controlled environment with fixed calibrated sensors. However, outside of these restrained conditions, it is often a preliminary step the rest of any reconstruction pipeline would heavily depend on. Most of multi view reconstruction are done from a set of images captured with a same sensor from different positions. An accurate estimate of the camera pose becomes then absolutely necessary and localization sensors such as GPS or Inertial Measurement Unit (IMU) are either cost prohibitive or often fail to provide this information with the required accuracy [5, 6] . Computer vision technics are therefore needed in order to refine these estimate. Optimizing camera pose estimate from image extracted features is often referred to as Structure from Motion and, or, Bundle adjustment.

Next, at the core of the art is shape or structure modeling. It can mean resolving the spatial state of the scene by being able to tell which part of the space is occupied by an object and which part is empty, or it can mean being able to accurately assign a three dimensional localization to each pixels on a 2D image from the scene. Under ideal conditions, (control environment, depth sensor), this problem is rather well understood and it is already possible to achieve highly accurate 3D rendering of objects that can then be used to benchmark other methods. It becomes a lot more challenging without depth sensor using only RGB images, from a single camera with complex motion. It is arguable that the 3D information is retrievable uniquely from RGB images in most cases, even if that requires heavy manual intervention, but this is most certainly more challenging than with the use of an active sensor. Active

sensors being less affordable, there is therefore an interest into solving the problem of 3D recontruction from RGB images.

In general shape modeling is a lot more challenging if the data comes from a passive sensor, and the choice of active methods is motivated by a will to achieve the best accuracy on that aspect.

Finally novel view generation consists in predicting a photorealistic rendering of a scene from a point of view where no image was taken. In addition to the almost unavoidable shape estimation, appearance modeling is needed. LIDAR although highly accurate for structure modeling is not sufficient in that case and RGB imaging is needed. A lot of challenges can arise due to illumination, reflections, shadows, changes in a living scenery.

As previously stated, we can distinguish different type of sensors, mainly separated in two categories: depth sensors and light intensity sensors. It is also possible to try to combine both type of sensors. One of the main challenges to tackle is the registration between the two type of data: It is indeed difficult to establish correspondances between features from depth images and features from RGB images, as the descriptors that are being computed in general cannot be expected to be similar. [7, 8]

Finally, the design of the whole reconstruction pipeline is often guided by the type of data we are trying to build a 3D model of. As Furukawa et al[9] stated it, we can distinguish three categories:

- Objects: In the simplest case, a single object is being reconstructed. This allows a variety of approaches based on segmentation.

- Scenes: A scene is composed of several objects making the segmentation more difficult.

- Crowded scenes: A complex scene with the addition of moving distractors. Structure from motion techniques cannot estimate with precision the position

of an object that would have moved between the time of acquisition of the two images. Image segmentation is unreliable.

From this categorization we can picture a wide range of scenarii, from an ideal situation when we are trying to reconstruct a single object in an ideal controlled environment, with fixed calibrated camera and or depth sensors or both, perfectly known and refined sensor pose estimate, and simplified background facilitating the segmentation of the targeted object in the images, to the most challenging one, when the sensor is constantly moving at high velocity, and the scene is complex and constantly changing. 3D reconstruction in Wide Area Motion Imagery from RGB images, that we will present in the next section, falls at the most challenging extreme of that spectrum.

1.1 3D Reconstruction in Wide Area Motion Imagery

This thesis works focuses on 3D reconstruction in Wide Area Motion Imagery also simply referred to as WAMI. It is a type of imagery taken from airborne camera, either an helicopter, plane, or drone, from high altitude, covering an area typically at least 1 square kilometers large (see figure 1.3 as example). WAMI based Remote sensing receives a lot of interest. It uses a camera with a large field of view, or sometimes an array of cameras, and offers the possibility to detect and track a lot of urban activity simultaneously with a single sensor unit. It can be used to enhance data from full motion video sensors. This saves man power by allowing each operator to cover more area. The images resolution is usually over a million pixels, and vehicles are therefore identifiable at this level of detail. This opens the door to efficient surveillance, traffic monitoring, mapping and change detection for example for disaster response after hurricanes and tornadoes or insurance assessment. However,

the amount of information to process is overwhelming for a human analyst. This motivates the development of algorithms to help them in that task, more particularly, vehicle detection and tracking and possibly at higher level event detection [10, 11, 12]. Image stabilization is a often a critical part of any of these algorithms but the rapid change of position and angle of view between frames makes this exercise particularly difficult. It is however a necessary task that can be made more accurate with the help of a 3D model. 3D data also allows to predict occlusion and rule out spurious matching.

1.1.1 From 2D Registration to 3D Reconstruction

2D registration computes an affine homography between consecutive images and or between each image and a reference map. The idea is to collect a set of images from an airborne camera and to build a mosaic. Being able to register each pixel from each image to a common coordinate frame is a critical part of tracking and change detection. If this is theoritically possible, 2D approache only gives decent results in very specific and restrained conditions. The complexity of an urban scene, or the variation of elevation are factors that make 2D registration often coming short or even unusable. This motivates the development of more complex algorithm using 3D modelization. (see figure 1.4)

1.1.2 Challenges

3D reconstruction in WAMI comes with many challenges of different nature [13]. Some are related to precision of the estimate of physical parameters modeling the camera position and intrinsic properties, and some are inherent to the approach that consists in trying to match the appearance of an object from different points of view.

Figure 1.3: An example of WAMI data: an image of downtown Columbia MO, at the edge of the campus. A Mizzou landmark, the columns near Jesse Hall can be seen on the botton corner.

precision requirement

The first challenging aspect coming to mind in remote sensing is the level of precision needed while estimating a wide range of parameters. In a typical set up, we are trying to localize a specific point in space with an error that reasonably should not exceed 1 meter, using information from a sensor located several thousands of meters away. At this distance a deviation of more than a thousand of a radian between the real light ray and the computed one would be sufficient to exceed that error bound. In fact, several sources of error are combining:

- error in positioning of camera center
- error in orientation of the camera
- error in localization on the image (due to razterization or just unperfect automatic detection)

Figure 1.5 illustrates how the different camera pose estimate errors combine.

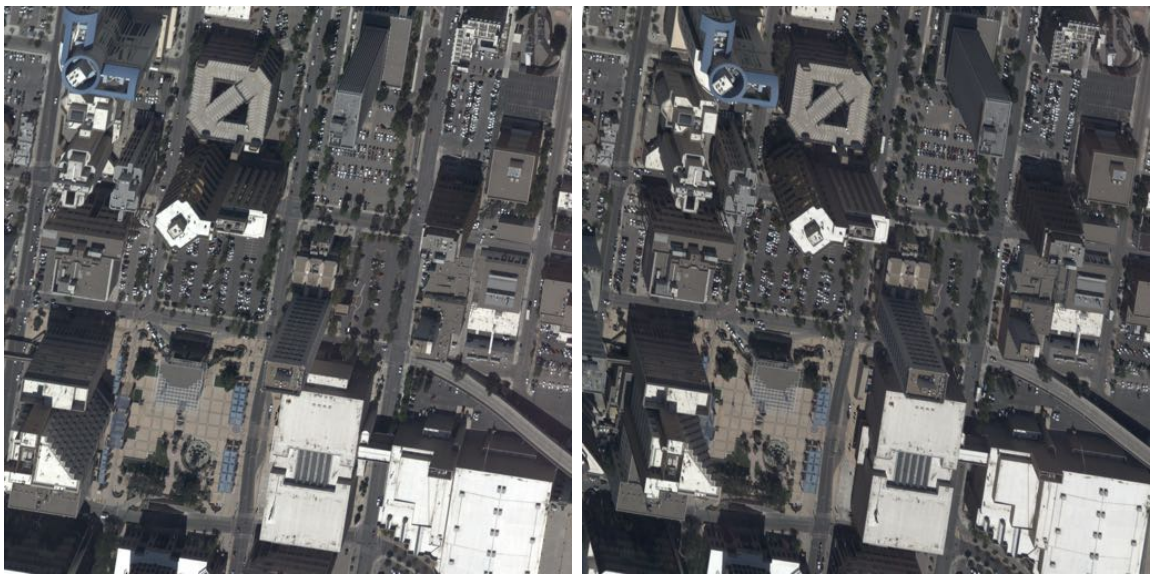


Figure 1.4: Example of motion parallax, side by side comparison of two different orthorectified views from Albuquerque Dataset, showing an obvious change of orientation of the tall buildings projections. .

It is actually impossible to get the required level of precision only with sensor data like GPS or IMU, for several reasons. The intrinsic precision of the sensor already gives little margin of error, since most GPS system precision are not better than a meter. But to make matter worse, there is a particularly difficult chalenge in synchronizing all the sensor data: image acquisition, position and angle reading that are expectedly taken from three independant sources [5].

As a result, Bundle Adjustment is an absolute necessity without which no accurate reconstruction is possible, and is integrated in most of pipelines attempting to reconstruct urban scenery [14, 15, 16].

As initially stated by Bhotika et al [17, 18] and more specifally illustrated in the context of WAMI data by Crispell[19], the sources of difficulty in 3D reconstruction can be separated into two general classes:

- scene ambiguity
- scene uncertainty

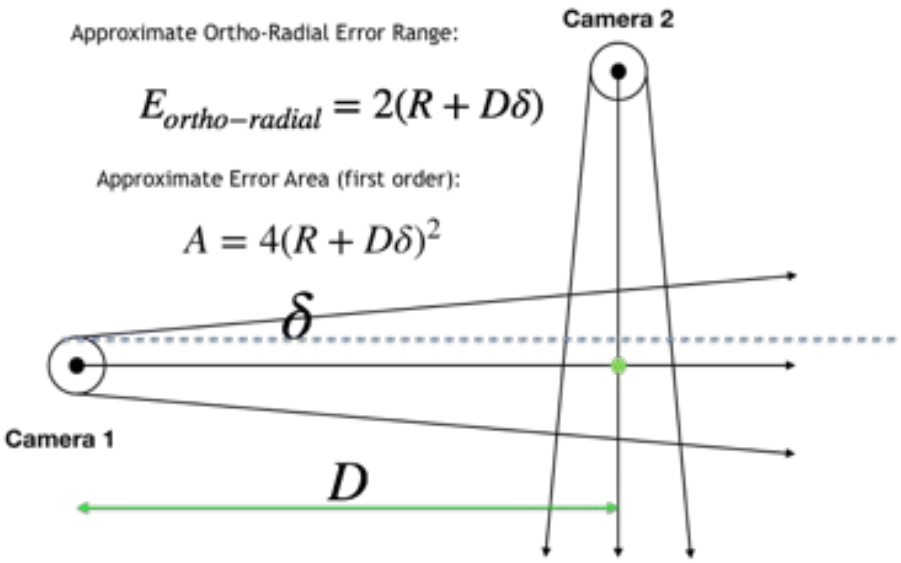


Figure 1.5: The camera position is known with a certain radius, and the ray orientation within a certain range of deviation. These two errors combine when trying to estimate the position of the ray intersection.

scene ambiguity

Scene ambiguity refers to the fact that several different 3D reconstruction can be photoconsistent. It means that the information contained in the imagery is not initself sufficient to retrieve the true geometry of the scenery and that the use of a prior or additional knowledge is needed to make a decision between different photoconsistent hypothesis. This is certainly even the case for the human brain, and this how we are able to create optical illusion. As our brain is making all sort of assumptions from context, lights, shadows, shapes. Ultimately 3D reconstruction from visible spectrum info, based only on photo consistency, is an ill-posed problem as even the photo hull is photo consistent.

scene uncertainty

Scene uncertainty refers to the fact that even a perfect reconstruction cannot be expected to maintain photo consistency due to unpredictable scenes changes such as moving vehicles, illumination and reflection variations. This problem is handled by

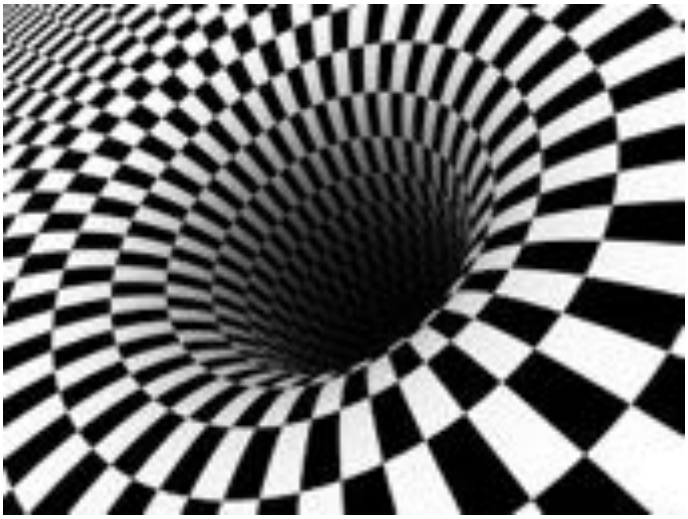


Figure 1.6: An illustration of 3D optical illusion (CopyRight: IllusionPoint.com [20]). Our Brain is extremely proficient at perceiving depth but is using assumptions that are not necessarily true and can therefore be tricked into seeing volume where there is none.

relaxing the photoconsistency constraint and working on quantifying the variance of the pixels intensity values. The lambertian assumption considers that the brightness of a surface is isotropic, and this would imply the same brightness in everyview. This is obviously never really the case, and a complex scene like a urban area makes the violation of this assumption particularly likely.

1.1.3 Contribution

This thesis focused on 3D reconstruction in Wide Area Motion Imagery (referred to as WAMI in the rest of this thesis). It is mostly aimed for surveillance but could find application in urban planning too for example. The algorithms proposed are only using RGB images and localization metadata. The contributions of this dissertation work can be listed as follows:

- Methods and experiments for evaluation of metadata precision
- Generation of synthetic data with simulated sensor noise for test and evaluation

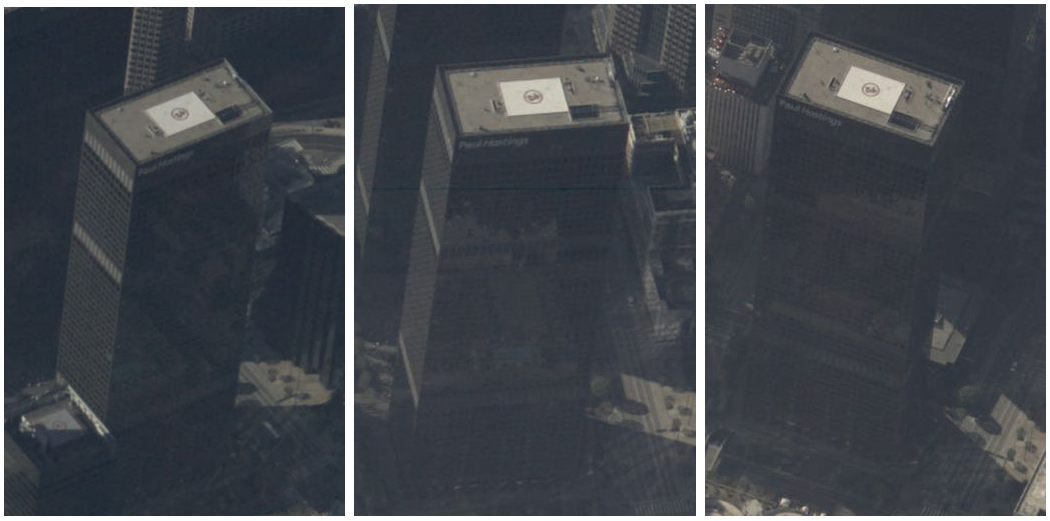


Figure 1.7: Highly reflective surfaces are not lambertian as their appearance dramatically change depending on the angle of view. Here is an example where the reflection of the swimming pool translates from one edge of the building facade to the other as the camera moves.

purpose

- Implementation of voting module and plane sweeping module
- Design and implementation of post processing module
- Design and implementation of point cloud coloring algorithms

We also present different possible use of the generated data, such as filtering, coverage summarization and data compression.

We tested our pipelines on our own dataset, supplied by Transparent Sky inc and this thesis presents the results. The novelty of this dissertation work resides in the presented possible combinations of voting, post processing, plane sweeping, depthmap fusion, and coloring modules, applied to these specific aerial dataset.

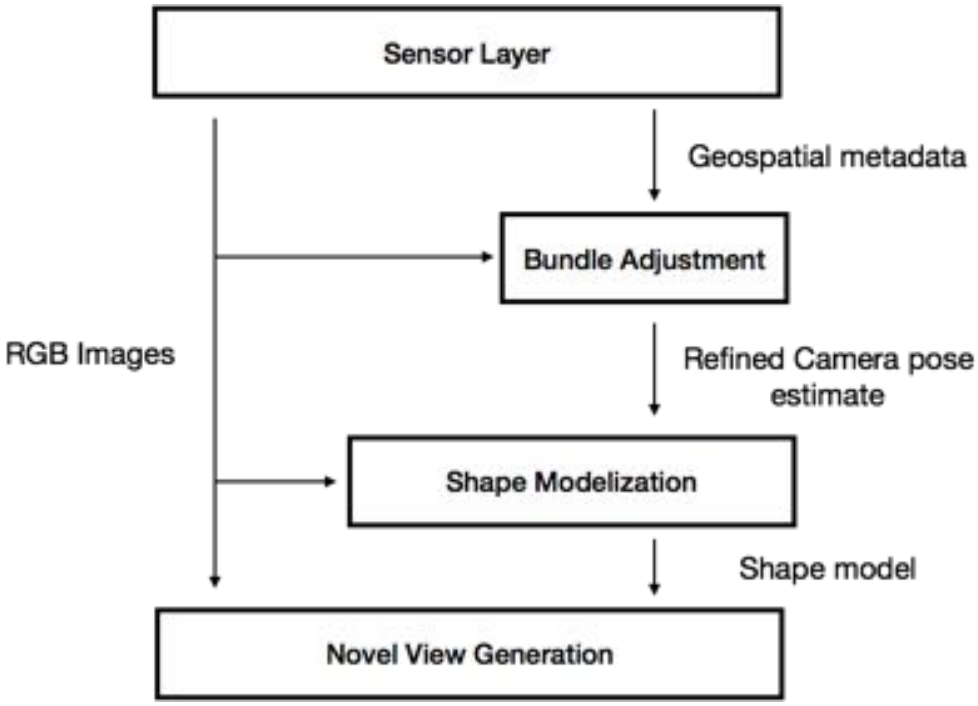


Figure 1.8: Block diagram of our proposed pipeline..

1.2 Datasets Overview

We experimented on data provided by Transparent Sky [71], a company based in New Mexico, who developped a camera system called Red river.

In addition to high resolution images, synchronized metadata are also available, giving for each image an estimate of the camera position and orientation, as well as the position of the object it is pointing at.

We generated point cloud for datasets:

- Four Hills
- Albuquerque
- Berkeley
- Los Angeles
- Columbia Missouri

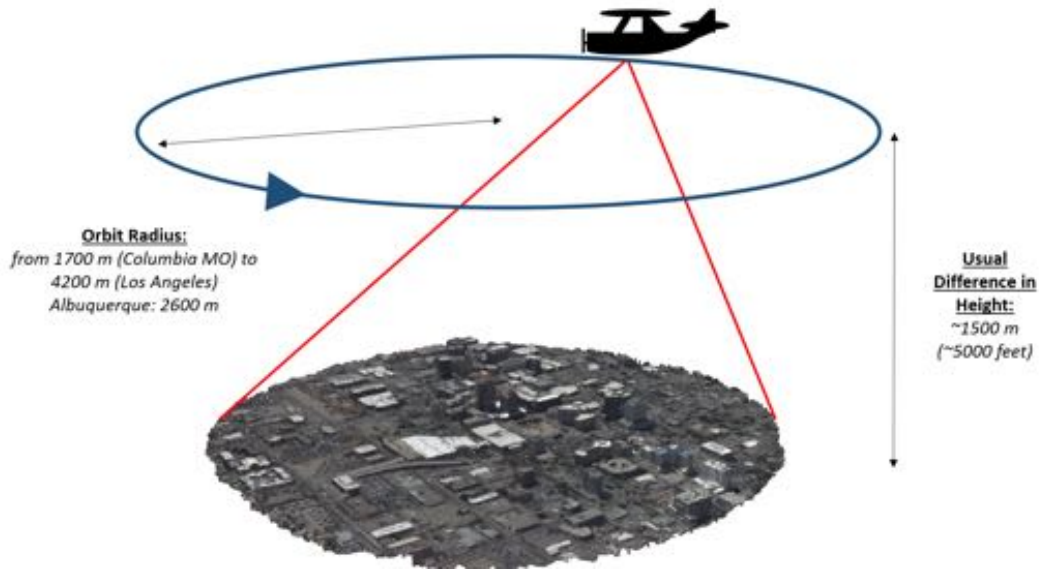


Figure 1.9: High resolution images (6600x4400 Pixels) and their synchronized metadata are acquired from airborne camera, GPS and IMU sensors. The airplane usually flights 1500 meters above the ground and complete an orbit in between 4 and 5 minutes, recording 4 frames every second. The area covered on the ground with complete overlap has an aproximate 1200 meters diameter.

The following of this section contains in section 1 a presentation for each dataset of a summarizing tab and some examples of images taken from the on-board camera and an evaluation of the metadata accuracy before and after bundle adjustment. Then the next section presents the a protocol to evaluate the sensor metadata and its conclusions.

1.2.1 Four Hills

Four Hills is one of our oldest datasets and is particularly challenging due to significant variations in elevation. The images were taken with an older camera compared to the other more recents datasets. This is why the image resolution and frame rate are lower. The images are taken from relatively closer to the ground too.

See Figure 1.10 for examples of frames.

Dataset	Four Hills
Orbit Radius (m)	1620
Difference in Height (m)	915
Number of Images One Orbit Full rate	80
Number of Images used for reconstruction	100 (Full rate, Overlap)
Elevation range (m)	100

Table 1.1: Four Hills summary

(a) *Frame 0* (b) *Frame 7* (c) *Frame 14* (d) *Frame 21*

(e) *Frame 28* (f) *Frame 35* (g) *Frame 42* (h) *Frame 49*

(i) *Frame 56* (j) *Frame 63* (k) *Frame 70* (l) *Frame 77*

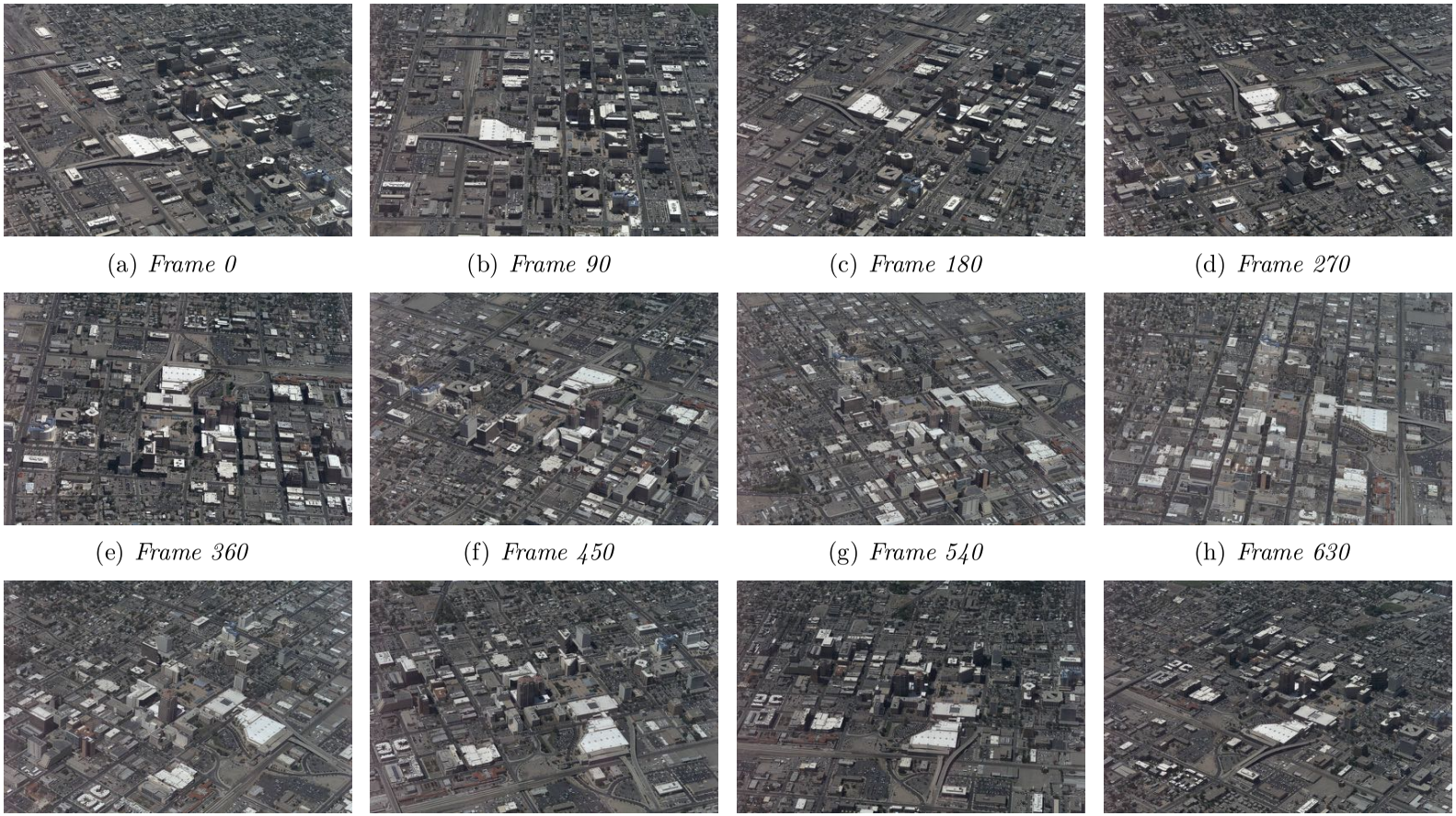
Figure 1.10: 12 frames from one orbit of Four Hills Dataset, spread around one flying orbit

Dataset	Albuquerque
Orbit Radius (m)	2589
Difference in Height (m)	1551
Number of Images One Orbit Full rate	1071
Number of Images used for reconstruction	216 (DownSampled)
Elevation range (m)	130

Table 1.2: Albuquerque Hills summary

1.2.2 Albuquerque

Albuquerque is a good baseline dataset. The center of the area contains the plaza surrounded by tall buildings, while the exterior ring is flatter but with some variations of elevation. There are only small areas with trees. Compared to the precedent dataset it has a higher frame rate which allows to use more images in a single orbit and the camera resolution is higher, allowing higher level of detail even from higher flight altitude.



(a) *Frame 0* (b) *Frame 90* (c) *Frame 180* (d) *Frame 270*

(e) *Frame 360* (f) *Frame 450* (g) *Frame 540* (h) *Frame 630*

(i) *Frame 720* (j) *Frame 810* (k) *Frame 900* (l) *Frame 990*

Figure 1.11: Twelve frames from one orbit of Albuquerque dataset, spread over one flying orbit.

Dataset	Berkeley
Orbit Radius (m)	1955
Difference in Height (m)	1750
Number of Images One Orbit Full rate	877
Number of Images used for reconstruction	220 (DownSampled)
Elevation range (m)	80

Table 1.3: Berkeley summary

1.2.3 Berkeley

Berkeley is a Dataset similar to Albuquerque with some building with flat homogeneous roofs, or a stadium field, and some variations in the elevation. Compared to Albuquerque, it has larger vegetation area too.

(a) *Frame 0* (b) *Frame 72* (c) *Frame 144* (d) *Frame 216*

(e) *Frame 288* (f) *Frame 360* (g) *Frame 432* (h) *Frame 504*

(i) *Frame 576* (j) *Frame 648* (k) *Frame 720* (l) *Frame 792*

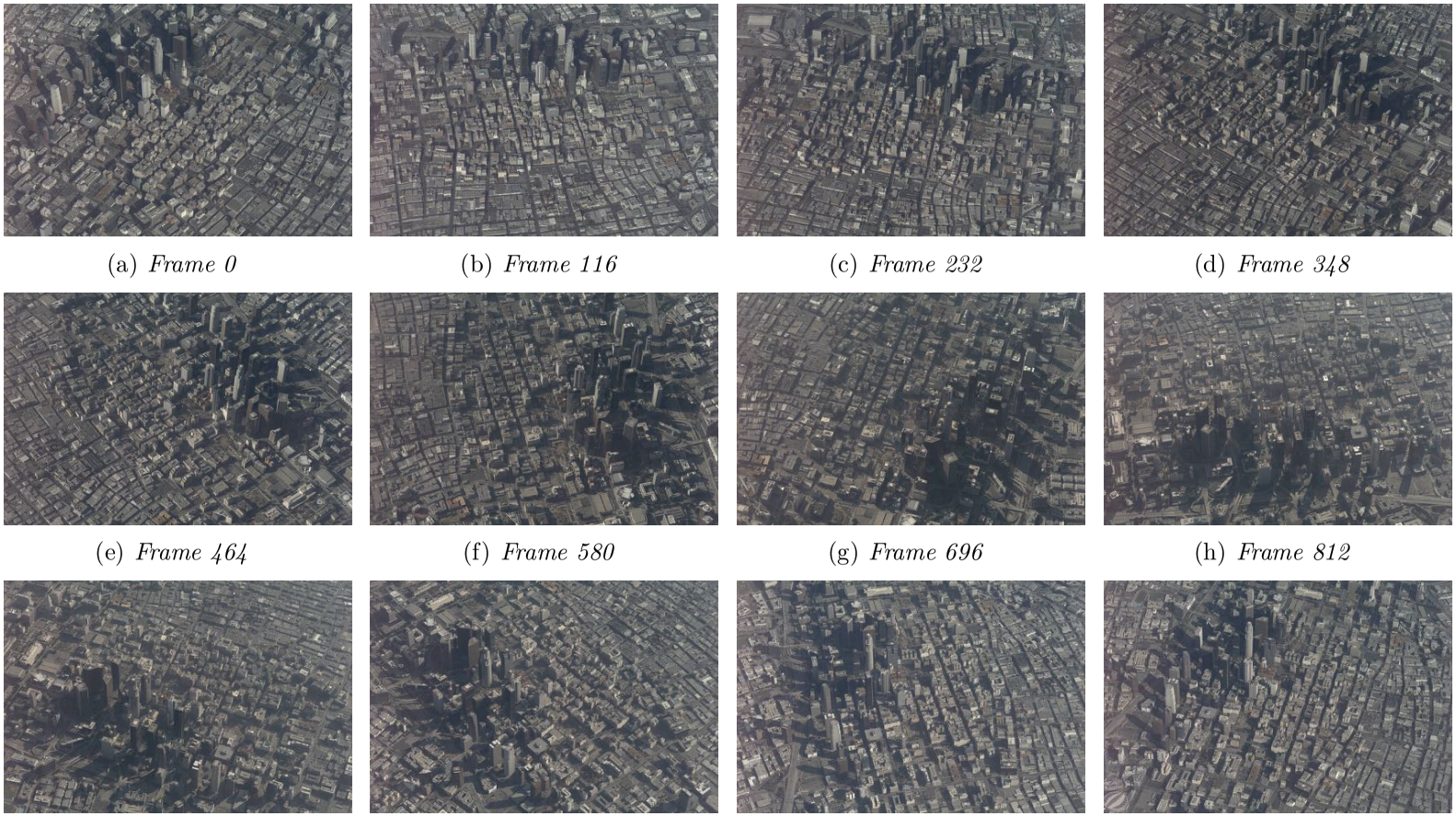
Figure 1.12: 12 frames from one orbit of Berkeley dataset, spread over one flying orbit

Dataset	Los Angeles
Orbit Radius (m)	4221
Difference in Height (m)	4081
Number of Images One Orbit Full rate	1400
Number of Images used for reconstruction	350
Elevation range (m)	350

Table 1.4: Los Angeles summary

1.2.4 Los Angeles

Los Angeles dataset contains images taken from a significantly higher altitude. Los Angeles has a relatively constant ground elevation, but a lot of tall buildings in the downtown area, occluding each other while the camera flies through the orbital path. The scenery is almost exclusively urban, except for a moderate size parc near the downtown area. Because of the high altitude and the wide range in elevation due to the tall buildings, Los Angeles is the most demanding dataset in term of storage and processing time.



(a) *Frame 0* (b) *Frame 116* (c) *Frame 232* (d) *Frame 348*

(e) *Frame 464* (f) *Frame 580* (g) *Frame 696* (h) *Frame 812*

(i) *Frame 928* (j) *Frame 1044* (k) *Frame 1160* (l) *Frame 1276*

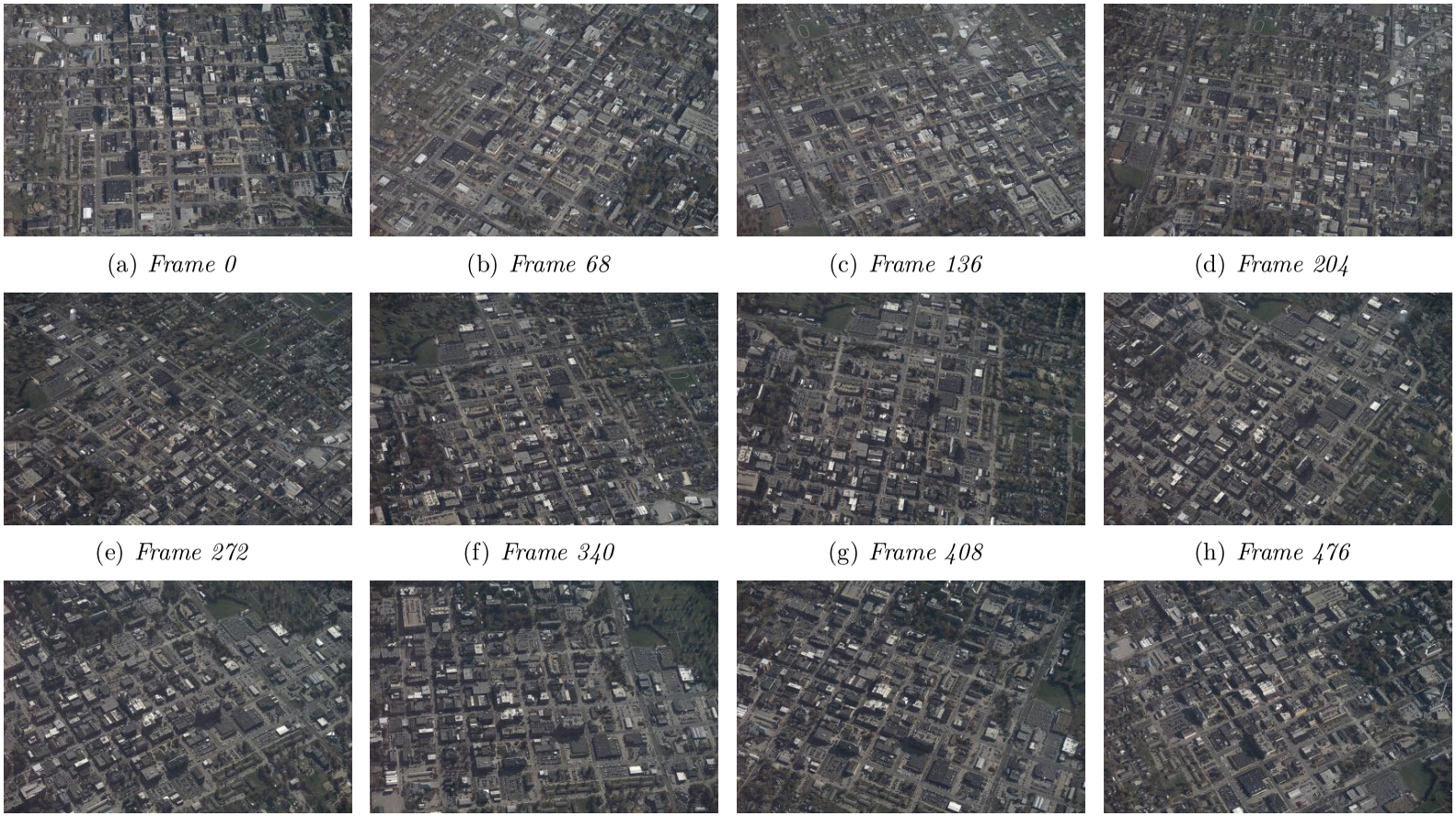
Figure 1.13: 12 frames from one orbit of Los Angeles dataset, spread over one flying orbit

Dataset	Columbia MO
Orbit Radius (m)	4221
Difference in Height (m)	2022
Number of Images One Orbit Full rate	805
Number of Images used for reconstruction	202
Elevation range (m)	120

Table 1.5: Columbia summary

1.2.5 Columbia

Columbia like Four Hills has a lot of variation in ground elevation with fewer tall buildings than Los Angeles or even Albuquerque. One of the focus area is the center of the campus, with the peace park, the columns. The scene is a good combination of urban area with streets and buildings, and parks with vegetation



(a) *Frame 0* (b) *Frame 68* (c) *Frame 136* (d) *Frame 204*

(e) *Frame 272* (f) *Frame 340* (g) *Frame 408* (h) *Frame 476*

(i) *Frame 544* (j) *Frame 612* (k) *Frame 680* (l) *Frame 748*

Figure 1.14: 12 frames from one orbit of Columbia Dataset, spread over one flying orbit

1.3 Dataset Preprocessing

1.3.1 Using Sensor Metadata

Assuming ideal detection and frame to frame matching, which we enforced in our experiment by manually tracking a small set of points through the image sequence, the smallest box that would be traversed by every ray would need to be 5 meters large if the camera pose estimate is solely based on the sensor data. After running our bundle adjustment pipeline, this error margin falls below 0.5 meter.

However, metadata can still be used for several purposes:

- Accelerating Bundle Adjustment convergence: The search of an optimum can be facilitated by initializing near the wanted solution. This is especially true in the case were several local extrema are possible.
- Estimating prior bounding box for the reconstruction

Description of the metadata evaluation protocole

In order to evaluate the accuracy of the metadata, we manually track a perfectly localized point through the whole image sequence. We chose the base of a lamppost as the localization of its visible surface stays relatively constant from any angle (See Figure 1.15).

Once we have the pixels coordinate of the considered object in every image we can reverse the projections equation to calculate the ray trajectory and then the projection on the ground. (See figure 1.16). The result is shown in Figure 1.17.

The ray trajectories can be visualized several ways (Figure 1.18 and 1.19). The conclusion is however always the same: the camera pose estimate from sensor measurement does not have the level of precision needed in order to hope to run any 3D reconstruction algorithm.

Figure 1.15: The lamppost is tracked manually through the whole sequence. The localization of the visible pixel in space can be considered constant because of the small size of the lamppost base.

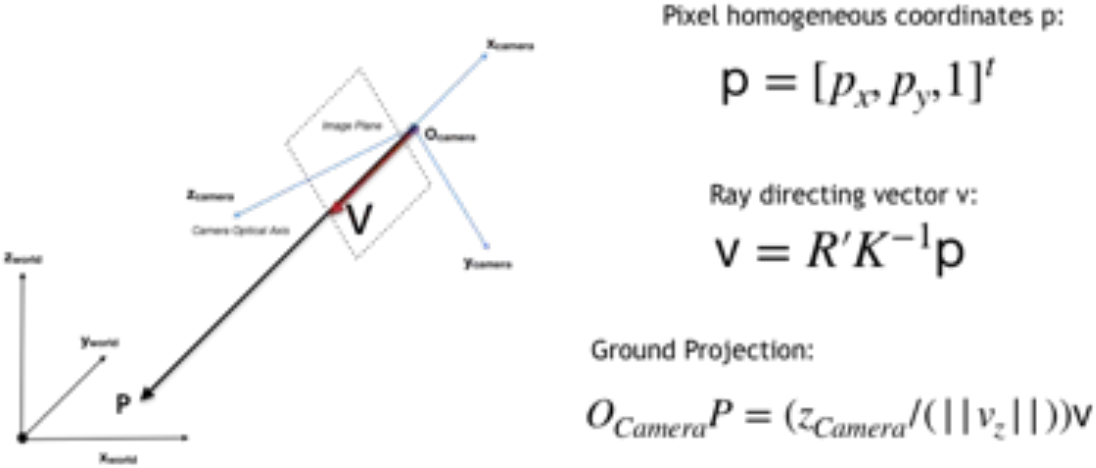


Figure 1.16: The projection matrix (More detailed formulation in Chapter 2) can be inversed to compute the ray direction, that is then intersected with the ground plane.

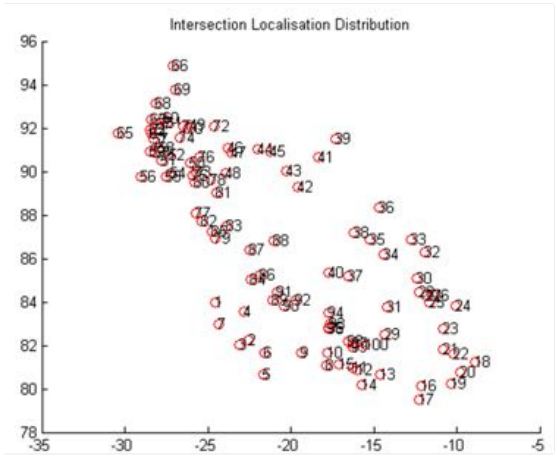


Figure 1.17: Illustration of Ground Projection metadata

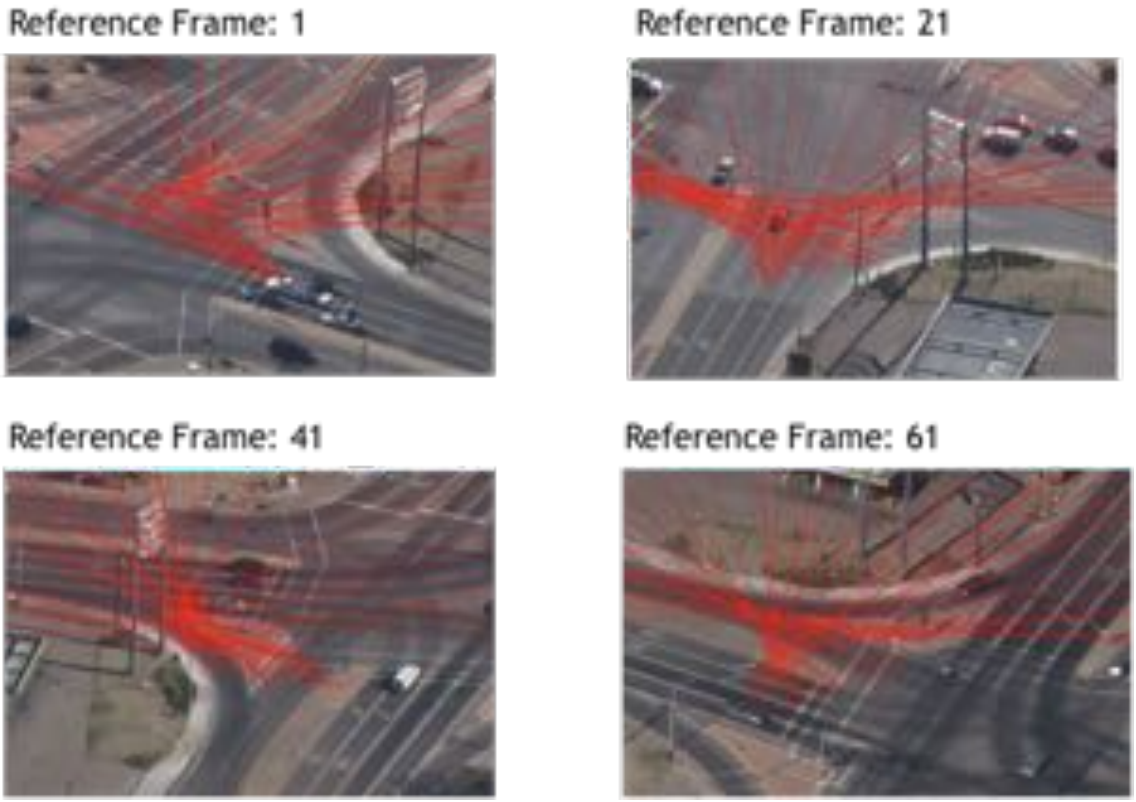


Figure 1.18: The rays supposed to aim at the lamppost base are missing the target by several meters.

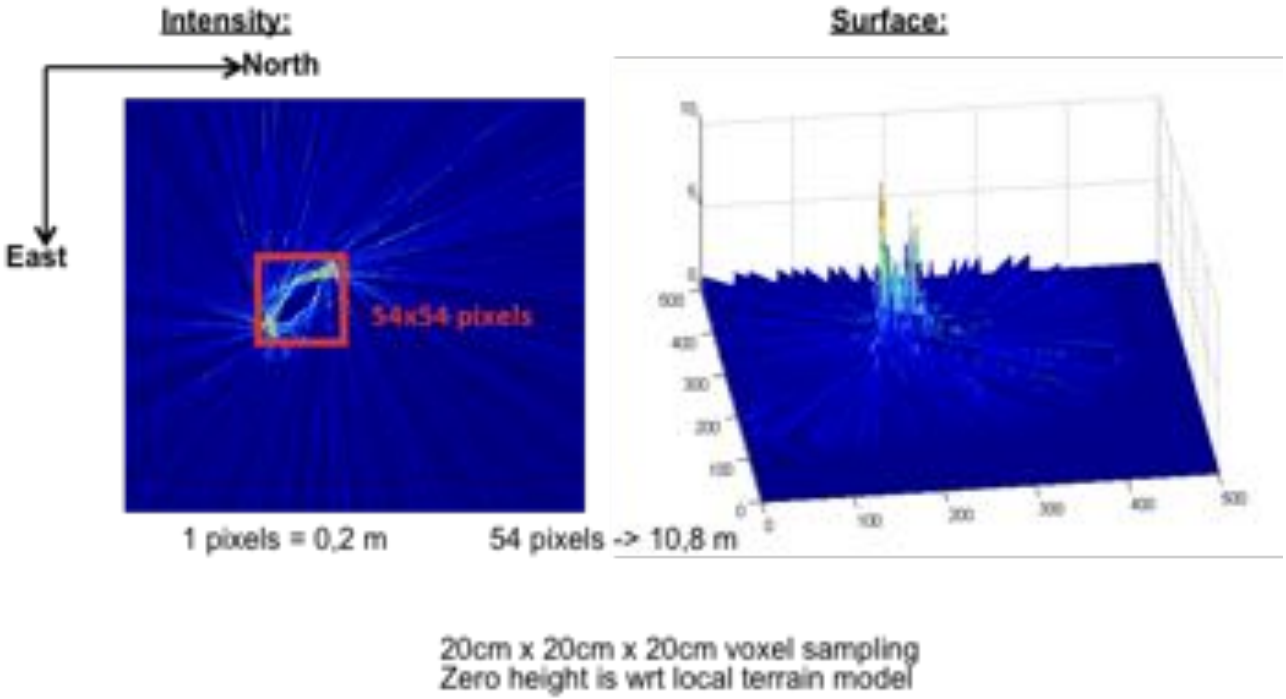


Figure 1.19: A top view projection of the rays aimed at the lamppost using sensor metadata.

In order to confirm that this type of convergence result should be expected we also ran simulations by generating synthetic data (See Figure 1.20) and adding gaussian noise modeling the expected precision of each measuring instrument. The result is shown in Figure 1.21.

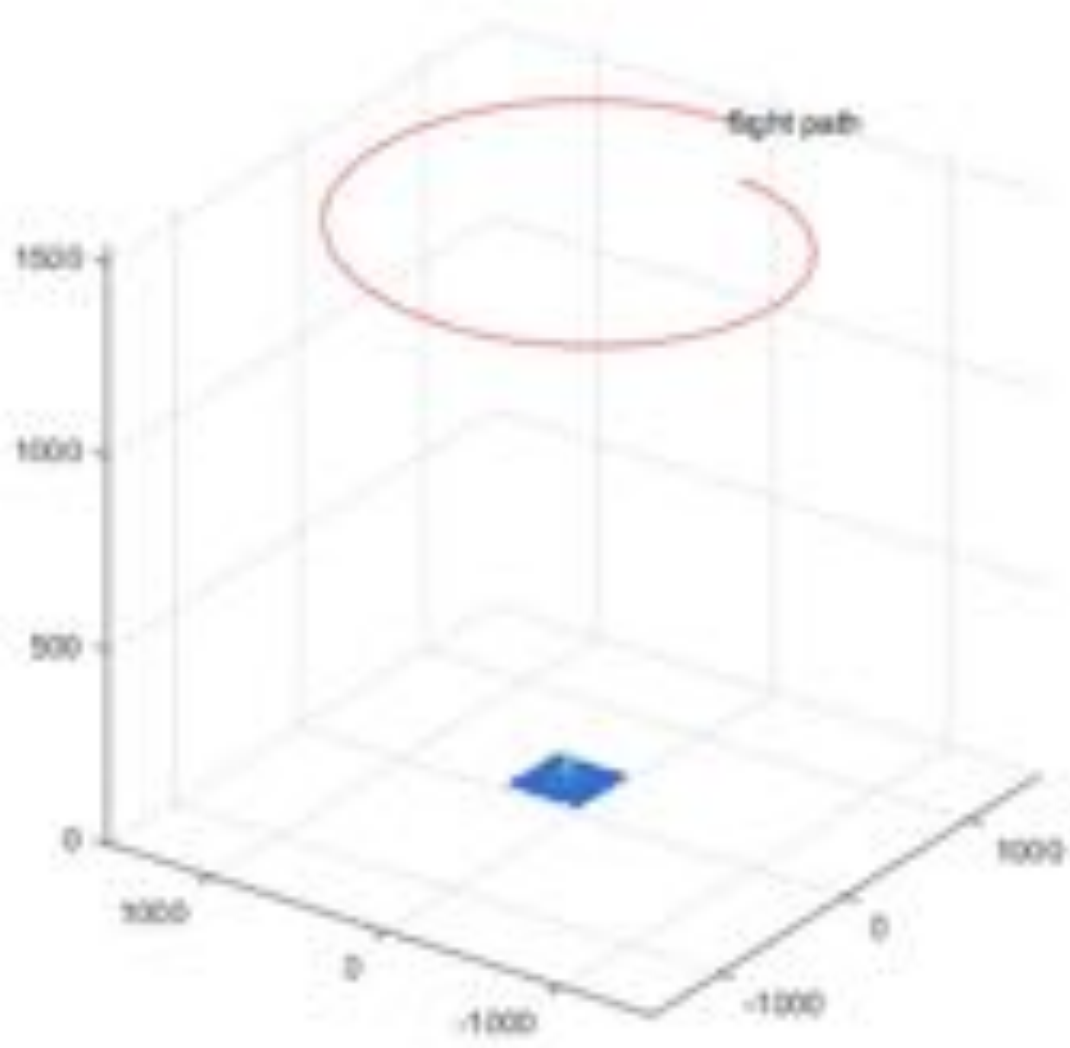


Figure 1.20: A flight path and a camera are generated syntheticaly. Different experiments with different noise parameters can be done in order to evaluate an error expectation.

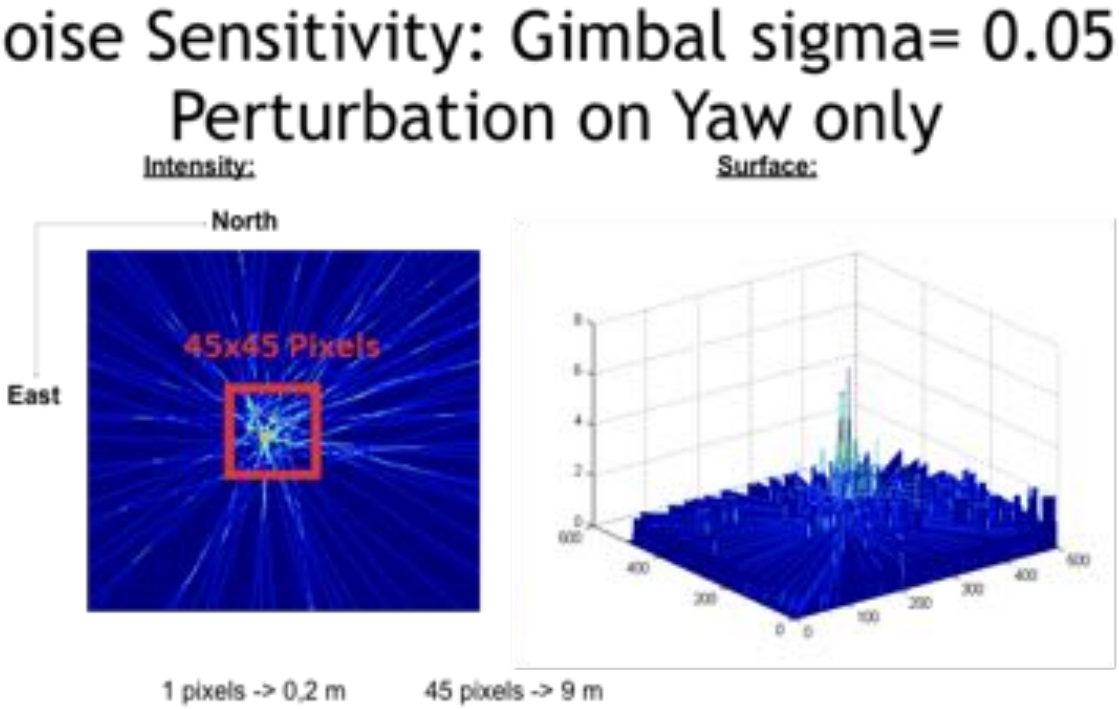


Figure 1.21: Introducing a gaussian angular noise of 0.05 degree is sufficient to obtain the same range of error observed in our experiments on Four Hills .

Conclusion

From this study we can confirm that bundle adjustment is a critical step in order to perform 3D reconstruction.

1.4 Thesis Outline

The rest of the dissertation is organized as follows:

- Chapter 2 presents fundamental concepts in 3D computer vision that are constantly used throughout the rest of the thesis
- In Chapter 3 we review related work and justify our contribution relative to the state of the art methods
- In Chapter 4 , we present the original voting pipeline, its advantages and limitations and all the experiments we tried with more or less successful outcome.
- Chapter 5 presents the plane sweeping pipeline, how it was motivated, and how it can take advantage of the work from the voting pipeline
- Chapter 6 discusses the difficulty of qualitatively evaluating 3D reconstruction of urban scene and proposes different strategies for evaluation and benchmarking
- Chapter 7 presents how the data we generated can be transformed for different applications, in particular for tracking and filtering.
- Chapter 8 concludes with a summary of our findings, what is left to solve and proposes possible extensions.

1.5 Notations

This thesis adopts standard conventions for mathematical formulas. Vectors and Matrices variable names are written in bold, scalars are in regular font. When vectors and matrices are written as tabulars of coefficient and that several frame of coordinates can be used, in order to improve clarity, the frame of coordinates are specified as indices. For example if the vector $\mathbf{v}$ is expressed by the triplet of coordinates (x_1, y_1, z_1) in the vector base B_1 ($\mathbf{e}_{x_1}$,$\mathbf{e}_{y_1}$,$\mathbf{e}_{z_1}$), we can write:

$$\mathbf{v} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}_{B_1} \tag{1.1}$$

If $\mathbf{v}$ is expressed by a second triplet of coodrinates (x_2, y_2, z_2) in a second vector base B_2 ($\mathbf{e}_{x_2}$,$\mathbf{e}_{y_2}$,$\mathbf{e}_{z_2}$), we name $\mathbf{R}_{B_2B_1}$ the change of basis matrix from B_1 to B_2:

$$\mathbf{v} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}_{B_2} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}_{B_2B_1} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}_{B_1} \tag{1.2}$$

Which is equivalent to:

$$\mathbf{v} = x_1\mathbf{e}_{x_1} + y_1\mathbf{e}_{y_1} + z_1\mathbf{e}_{z_1} = x_2\mathbf{e}_{x_2} + y_2\mathbf{e}_{y_2} + z_2\mathbf{e}_{z_2} \tag{1.3}$$

Chapter 2

Fundamentals of 3D Computer Vision

This chapter gives an overview of the basis for 3D computer vision that are being systematically used including in this dissertation. Some Comprehensive theory formalization books are also available [21, 22].

2.1 Pinhole Camera Model

The Pinhole Camera Model (Figure 2.1) allows to easily calculate the coordinates of the projection of a point from the three-dimensional space onto an image plane. The camera is considered as ideal on many aspect, lenses distortions and unfocused objects blurring phenomenon are ignored. In a real physical sensor, the light goes through the aperture and prints an inverse image on a plane located behind the point of entry. It is however more intuitive to use the virtual image plane and to consider the image is formed on a plane located between the object and the aperture, at a distance corresponding to the physical focal length of the camera. The usually rectangular boundaries of the visible frame, and the ideal pinhole or center of projection, are

forming what is sometimes called the pyramid of vision.

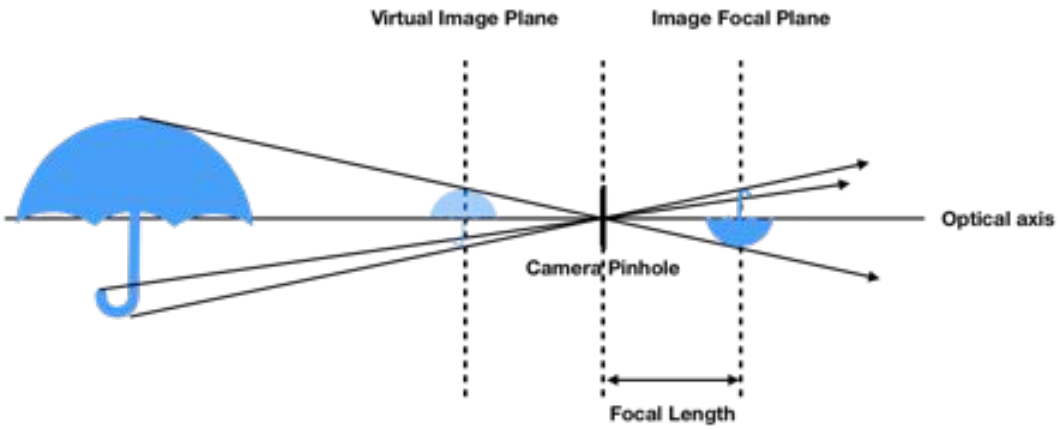


Figure 2.1: The pinhole camera model: In the ideal case, the light rays going through the camera pinhole are forming an inversed image on the Image Focal Plane. It is more intuitive and mathematically equivalent to consider that the image forms on the virtual image plane and thaat the camera pinhole point acts as a center of projection.

The camera extrinsic parameters can be modeled by a 3 dimensional vector **t** defining its position in space and a 3x3 rotation matrix **R** defining its axis orientation. In addition, a 3x3 matrix **K** summarizes its intrinsic parameters such as the focal length or radial distortions.

2.1.1 The Intrinsic Matrix K

In the ideal camera pinhole model, the intrinsic matrix **K** can be written has follows:

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

f_x and f_y are measuring the focal length in pixels. In the ideal case we have

$$f_x = f_y$$

, but this is however not always verified for several possible reasons:

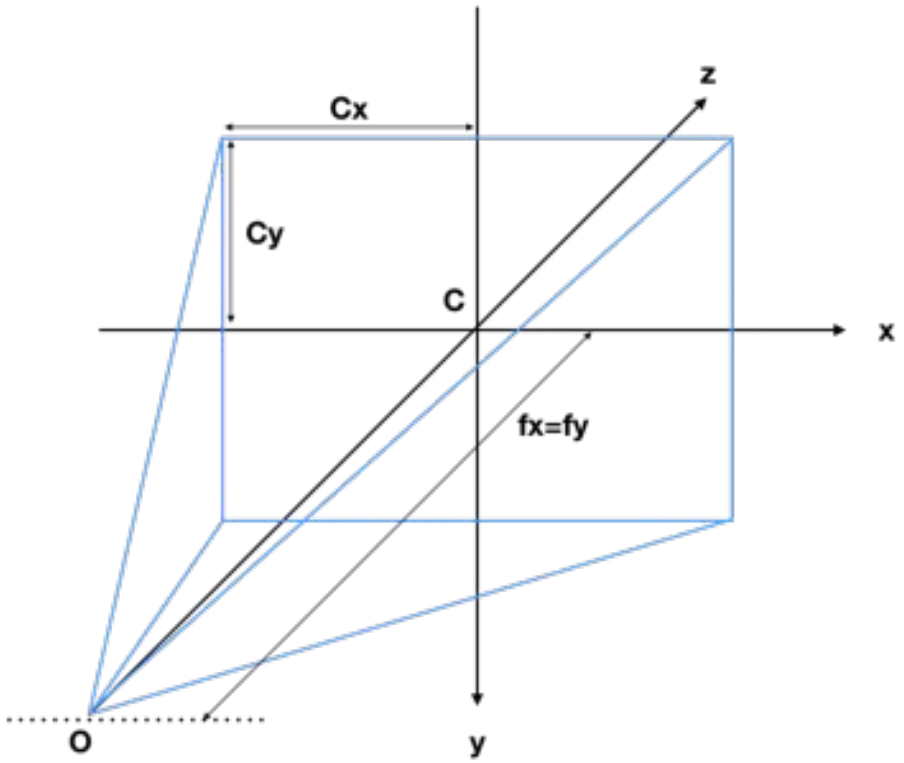


Figure 2.2: Aspect ratio defined by the values in intrinsic matrix **K**.

- Lens distorsion
- Distorsion introduced by post processing

C_x and C_y represent the translation between the Image origin and the optical axis. s can model what is called shear distortion, a phenomenon that is in general ignored in order to simplify the model.

Therefore, for an ideal pinhole camera model, giving a 6600x4400 image with ideal focal length parameters, and center offset, and with the a X:Y:Z aspect ratio equal to 6600:4400:17500, the intrinsinc **K** matrix would be as follows:

$$\boldsymbol{K} = \begin{bmatrix} 17500 & 0 & 3300 \\ 0 & 17500 & 2200 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

K represents the parameters depending only on the intrinsic properties of the

camera sensor. Once it has been estimated, its values can be stored and are expected to remain constant, unlike the extrinsic parameters representing the camera pose that can change over time as the camera changes position.

other distorsions

The camera matrix **K** does not model radial and tangential distorsions that tend to curve lines expected to be straight. These distorsion phenomenon can be separated into three types:

- Barrel distorsion: image magnification decreases as the distance from the optical axis increases.
- Pincushion distorsion: image magnification increases as the distance from the optical axis increases.
- Mustache distorsion is a complex combination of the two previous distorsions

It is possible to correct for these distorsion through image processing. For example this the division model initially proposed by Lenz [23]:

$$x_u = x_c + \frac{x_d - x_c}{1 + K_1 r^2 + K_2 r^4} \tag{2.3}$$

$$y_u = y_c + \frac{y_d - y_c}{1 + K_1 r^2 + K_2 r^4} \tag{2.4}$$

where $r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$ is the distance from the center

Another possible sort of distorsion is when the lense orientation is not parallel to the image plane. This is referred to as tangential distorsion. The model below

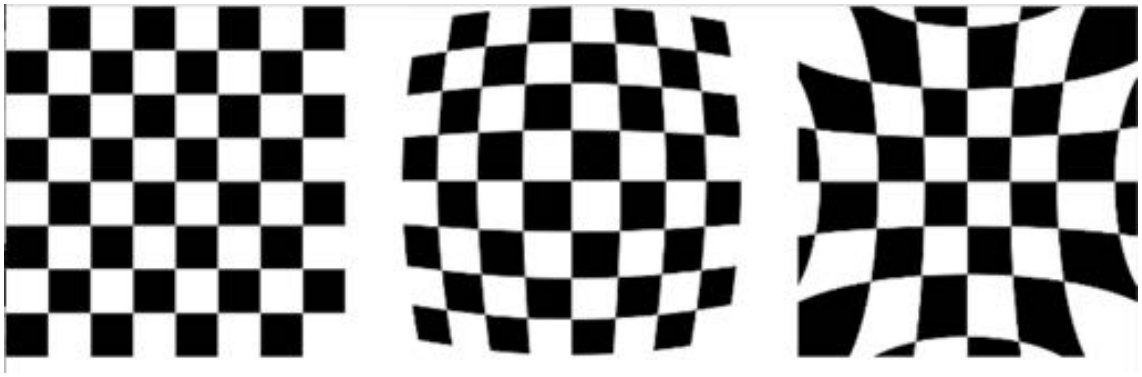


Figure 2.3: Left: No Distortion. Center: Positive Radial Distortion (Barrel) Right: Negative Radial Distortion (Pincushion)

accounts for both radial and tangential distorsions [24]:

$$x_u = x_d + (x_d - x_c)(K_1 r^2 + K_2 r^4 + ..) + (P_1(r^2 + 2(x_d - x_c)^2) + 2P_2(x_d - x_c)(y_d - y_c))(1 + P_3 r^2 + P_4 r^4 + ..) \quad (2.5)$$

$$y_u = x_d + (y_d - y_c)(K_1 r^2 + K_2 r^4 + ..) + (2P_1(x_d - x_c)(y_d - y_c) + P_2(r^2 + 2(y_d - y_c)^2))(1 + P_3 r^2 + P_4 r^4 + ..) \quad (2.6)$$

2.1.2 The Rotation Matrix R

The columns of the Rotation matrix are the coordinates of the camera axis vectors expressed in the reference frame.

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}_{Camera,World} \quad (2.7)$$

The camera frame of coordinate is set so that the X axis is colinear with the horizontal axis of the image plane, the Y axis colinear to its vertical axis, and the Z

axis colinear to the optical axis. $\mathbf{R}$ is therefore the change of basis matrix between the world and the camera frame of coordinates. Furthermore, $\mathbf{R}$ is an orthogonal matrix, and as a very important property, it can be inversed simply by transposing it:

$$\mathbf{R}^t * \mathbf{R} = \mathbf{R} * \mathbf{R}^t = \mathbf{I}_3 \tag{2.8}$$

2.1.3 The Translation Vector t

The translation vector is expressed in the camera system of coordinate:

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}_{Camera} \tag{2.9}$$

t_x, t_y and t_z are therefore the translation of the camera center from the world origin respectively along the image horizontal axis, the image vertical axis, and the camera optical axis (See figure 2.4) .

Let $\mathbf{P}$ be the position vector of the camera center of projection, expressed in the world frame of coordinates, the relationship between $\mathbf{P}$ and $\mathbf{t}$ is the following:

$$\mathbf{P} = \begin{bmatrix} P_X \\ P_Y \\ P_Z \end{bmatrix}_{World} = \text{-}\mathbf{R}^t * \mathbf{t} \tag{2.10}$$

2.1.4 Projection Matrix

$\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$ entirely define the camera pose estimate and allow to calculate for any point in the real world (x, y, z), the image coordinate (p_x, p_y) of the pixel it is projected onto. We define P_m the projection matrix as follows:

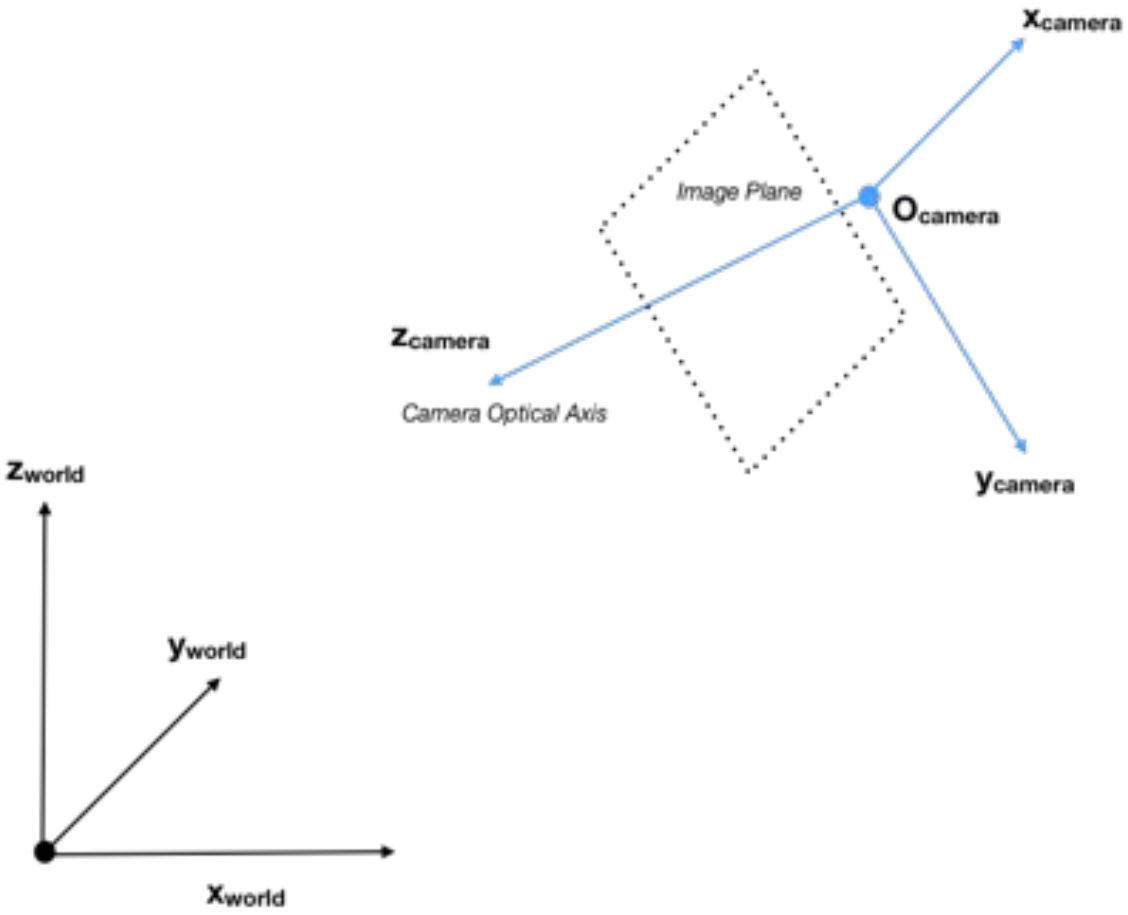


Figure 2.4: The Camera orientation is defined by three axis: thw two image axis x and y and the optical axis z.

$$\boldsymbol{Pm} = \mathbf{K} * \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \tag{2.11}$$

$$s * \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}_{Image} = \mathbf{K} * \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{World} = \boldsymbol{Pm} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{World} \tag{2.12}$$

The projection matrix is applied to vectors expressed in homogeneous coordinates.

2.1.5 Bundle Adjustment Formulation

The Bundle Adjustment problem is an optimization problem that aims to simultaneously refine the estimate of the camera pose and the 3D localization of the tie points used for the optimization.

The formulation is as follows: if we have N_{images} images, $N_{tiepoints}$ 3D points whose existence is inferred from matching descriptors from the image set. For point i and image j we use x_i to refer to point i vector of homogeneous coordinates, P_{mj} the projection matrix of camera j, and z_{ij} to refer to the observation of point i on image j and S_j the subset of points observed on image j. The bundle adjustment problem in that case consists in minimizing the following cost function:

$$C = \sum_{j=0}^{N_{images}} \sum_{i \in S_j} ||z_{ij} - P_{mj} x_i|| \tag{2.13}$$

2.2 Camera Calibration

The previous parameters can be estimated using camera calibration methods. A wide range of approaches have been developped over time, it generally consists in using a known pattern in order to retrieve the camera pose and the intrinsic parameters including focal length and lense distorsion. This pattern is typically a grid (or checkerboard) which enables reliable and accurate detection of lines and corners. All the camera calibration parameters can then be inferred from the difference between the identified corners on the image and their actual disposition in the real world, which is completely known. Camera Calibration being the first step towards inferring measurement from computer vision, it has been widely studied and several surveys of the different approaches are available [25, 26].

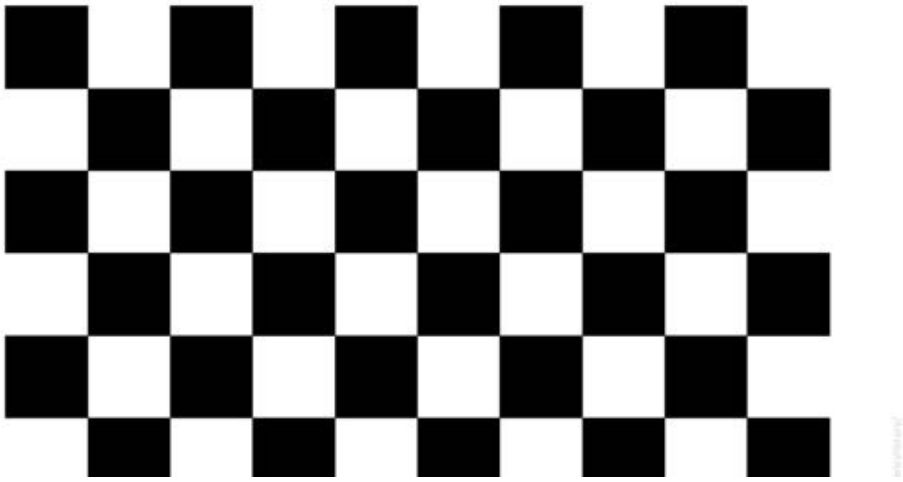


Figure 2.5: A checkboard used for camera calibration

2.3 Epipolar Geometry and Image Transformation

There are several ways to establish a relationship between two images of a same scene taken from different points of view. This is often needed to build a larger image than the camera physical limitation allows to, and in general required for example for background substraction.

2.3.1 Homography

The simplest way to register two images is by using an homography that warps the paired image to the reference image.

The most standard way to compute an homography follows the following steps:

- establish point correspondences between the two images (using descriptors for example like SIFT[28], SURF [29] or ORBS [30])
- use RANSAC with direct linear transform to find the homography that minimizes the number of outliers
- apply back transformation to warp the sensed image

Figure 2.6: After applying the homography estimated with Direct Linear Transform and RANSAC. The target image is warped and overlayed on the reference image that remains unchanged. [27]

Although it is still widely used, this transformation is not ideal to build a panorama from images with high variation of point of views. It somehow assumes near-planar structure of the scene and will perform poorly on a scene with complex 3D structure. This motivates creation of models taking better account of the 3D structure of the scene.

2.3.2 Epipolar Geometry

The epipolar geometry (See figure 2.7) describes the geometrical relationship between two views. It defines the following notions for a pair of images:

- The *epipole*: the projection of the camera center of projection onto the paired image plane.
- The *baseline* joins the two cameras center of projection.
- The *epipolar plane* is formed by the previously defined *baseline* and a point in

3D space. There are an infinity of epipolar planes for a specific pair of cameras, but they all intersect at the baseline.

- *Epipolar lines* go in pairs. On each image, they join the epipole to the point projection. It is as if we were visualizaing a light ray joining the paired camera to the point in 3D space.

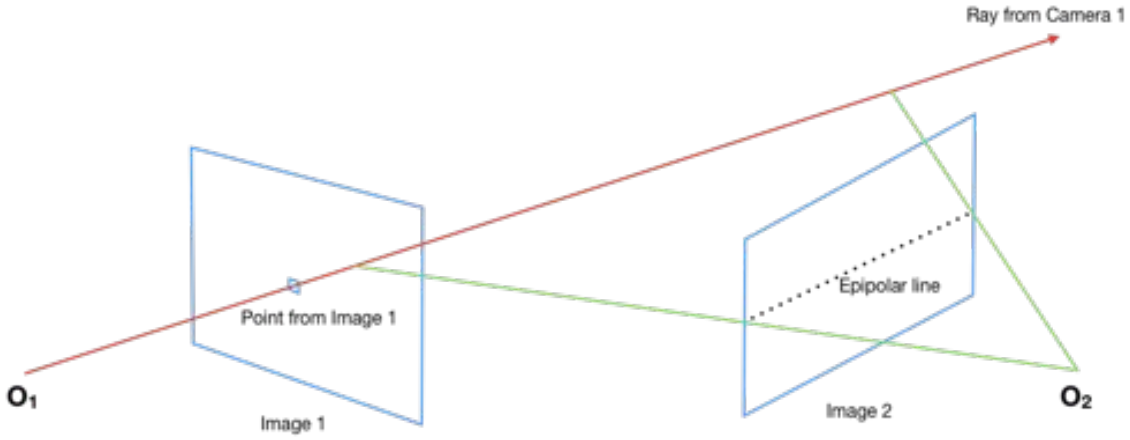


Figure 2.7: A point in one image is a line in the other one.

2.3.3 Fundamental and Essential Matrices

Fundamental and Essential matrices are representing the epipolar constraint, in an equation called the fundamental constraint:

$$\mathbf{y}^t * \mathbf{E} * \mathbf{x} = 0 \tag{2.14}$$

if $\mathbf{x}$ and $\mathbf{y}$ are points from two images, on each other epipolar lines.

The fundamental matrix is derived from the essential matrix with the following equation:

$$F = K'^t * E * K \tag{2.15}$$

relation with K,R and t

it is possible to deduce the fundamental matrix from the camera intrinsic and extrinsic parameters:

$$\mathbf{E} = \mathbf{R} \left[\mathbf{t} \right]_x \tag{2.16}$$

2.3.4 4 Points and 8 Points Algorithms

It is possible to estimate the fundamental matrix using point correspondances between the images. One of the algorithm requires 8 points correspondances between a pair of images. Just like for the estimate of an homography, RANSAC can be used to refine the fundamental matrix estimate and to deal with outliers for example when the method to obtain the correspondances is automatic and prone to spurious matches.

2.4 Data Representation

This section is an overview of the various formats that can be used to represent 3D Data. They can be mainly separated into three categories:

- Depthmaps
- Voxels grids
- Point Cloud and meshes

2.4.1 Depthmap

A Depthmap is a 2D image, storing spatial information. In its most standard use, for each pixel (px,py), the image would store at this location the distance between the camera center of projection and the nearest surface that project onto that pixel (see

figure 2.8). This can be the immediate output of a depth sensor, it is however often needed to register this data into another frame of reference.

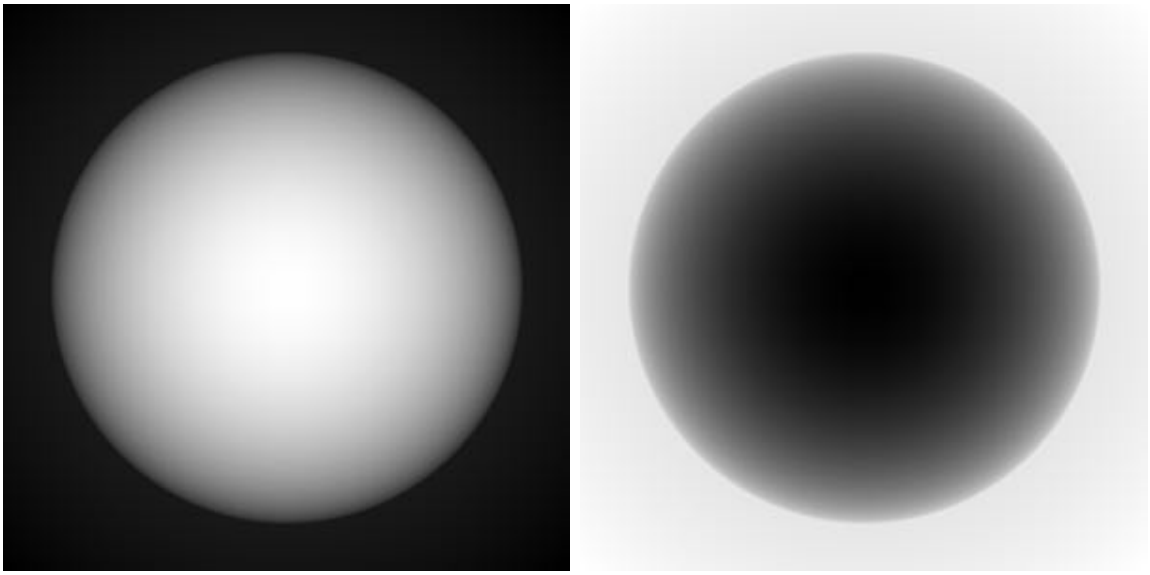


Figure 2.8: An example of Depthmap. The image intensity is used to represent the distance between the surface of a sphere and the camera center of projection. Two conventions are possible, nearest surface can be either lighter (left) or darker (right).

2.4.2 Voxel Grid

Another widely used data structure is the voxel grid. A voxel is the 3 dimensional generalization of a pixel. A voxel grid is therefore a regular sampling of the 3D space into subdividing cubes. The simplest way to use this structure in order to model the shape of an object is to build a grid all around the target object and to determine for each individual voxel whether it is occupied or empty. The quality of the shape rendering is then directly linked to the grid resolution. This data structure allow efficient computation by instant access to any loaction in space. However, if only a binary information on occupancy is maintained, this is rarely the most efficient way to store a model as keeping a list of the occupied voxels is often preferable. This is why this data structure is more often used as an intermediary model, usually

storing a measurement of the occupancy probability. After this probability space is thresholded, the use of point cloud or Mesh is often preferred.

Figure 2.9: An example of Voxel Grid on the Albuquerque Dataset. Because of memory limitation is is often impossible to cover the whole scene at high resolution.

razterization

One of the consequences of space discretization is that geometric objects such has lines or solids sometime need to be converted to the list of voxels they intersect or contain. The use of rays as a geometrical concept being ubiquitous in 3D recontruction algorithms, a 3D generalization of Bresenham is needed to get the list of voxels that are being traversed by a specific ray.

optimization

It is possible to optimize the implementation of some of these tasks, using appropriate data structure. Indeed, most of the space is empty and some compression strategies can be used in order to be more memory efficient or even to speed up some of the calculations. For that purpose, one powerful data structure is the octree. If the whole space is a cube, the tree recursively divides the space into 8 subparts. This allows to

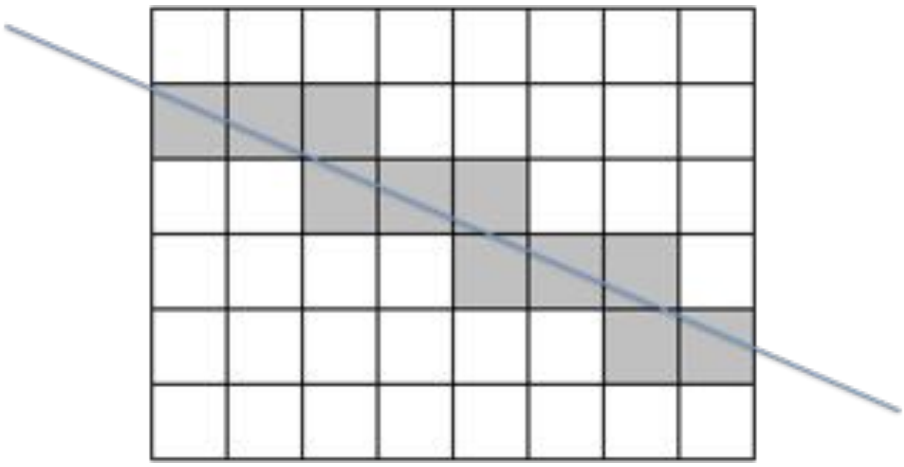


Figure 2.10: From the mathematical line, raszterization algorithms retrieve the set of discrete unit of space that are being traversed

take advantage of the heterogeneous occupancy

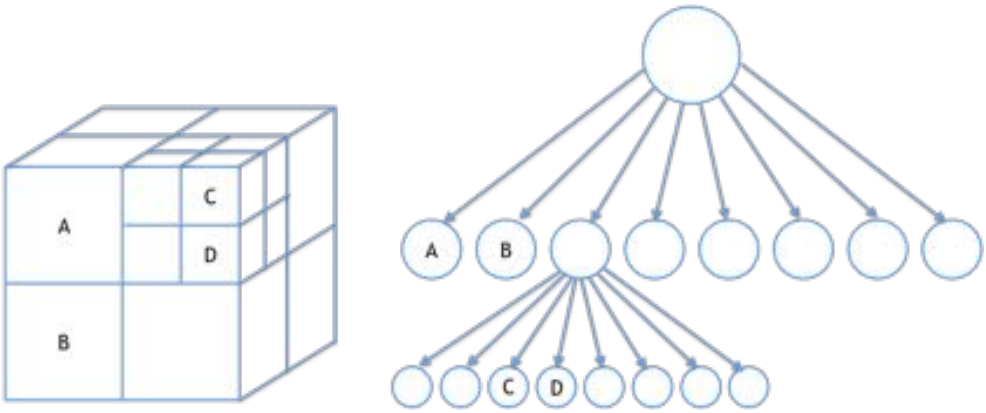


Figure 2.11: Illustration of an octree

2.4.3 Point Cloud and Mesh

Point Clouds are the preferred format for 3D model. In its simplest form it can just be a list of 3D coordinates, but is is possible to augment the data for example by assigning a RGB color to each point.

An even higher level of representation is reached with meshes. The base information is still a list of point but a relationship graph linking a point to its neighbors is

built. In the most common case, triplets of points are forming a network of rectangular faces. A color can then be assigned to each face, and for an even more realistic rendering, the face normal is used to model the light reflection.

delauney triangulation

Delauney triangulation is a triangle mesh construction from a set of discrete points. The specific property of this triangulation is that no point of the set is inside the circumcircle of any of the triangles. Except for a few degenerated cases, existence and unicity is garanteed if using the euclidian distance. The Delauney triangulation is closely related to the voronoi diagram and is defined as its dual. There are several algorithms allowing to construct the delauney triangulation, the simplest being the flip algorithms. Tools out of the box exist to quickly compute a mesh from a point cloud, based on this approach.

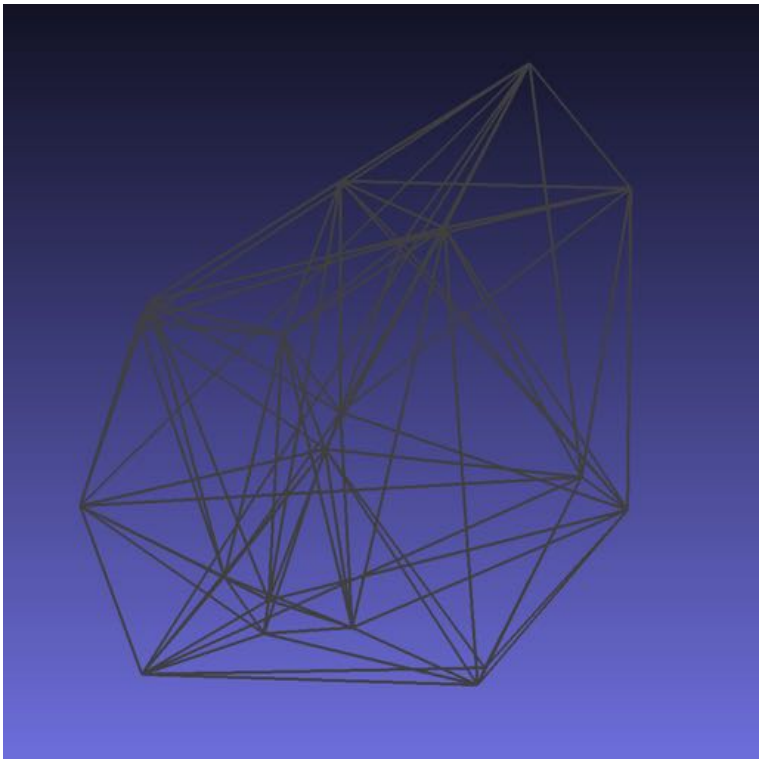


Figure 2.12: Delauney triangulation of a random set of points in Meshlab

normal estimation and plane fitting

Another problematic in point cloud processing is the estimation . of normal that can eventually be used in order to compute a surface, or for better rendering

Plane fitting can be done with several methods:

- Principal Component Analysis
- Ransac
- Hough Transform

The Hough transform, originaly a method to detect lines in images (Figure 2.13), can be generalized to plane detection in 3D and naturally infer a set of plane from an initially unstructured set of points. The two other methods can only extract a single plane from the set they are applied on. In the event of a complex scene, it is therefore needed to cluster the set of points first or to divide it into subsets using for instance Nearest Neighbors. A basic RANSAC approach would select three random poiont from the set and threshold the other points as outliers if they were too far away from the considered plane.

2.5 Data Visualization

Some of these data representation also pose interesting challenges in term of visualization. Mesh gives the representation that is the closest to the human perception of his environment but building a mesh is a non trivial task and it can sometimes deteriorate the accuracy of the structure model estimate. Point clouds on that aspect are safer in term of accuracy of what is being displayed, but less complete and require a very high resolution in order to achieve fidel rendering of a scene. On the opposite, surface fitting and meshes can allow to compress the data.

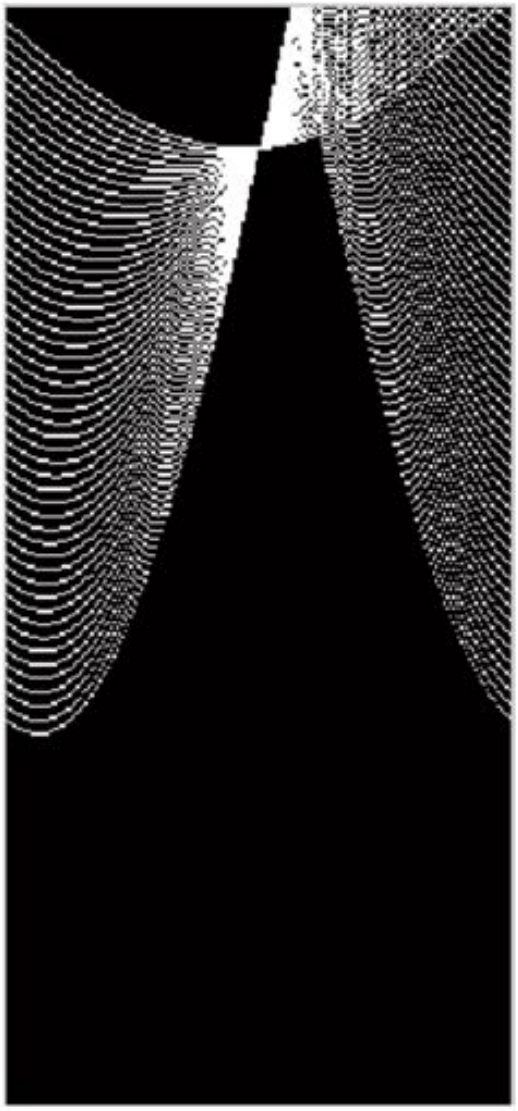
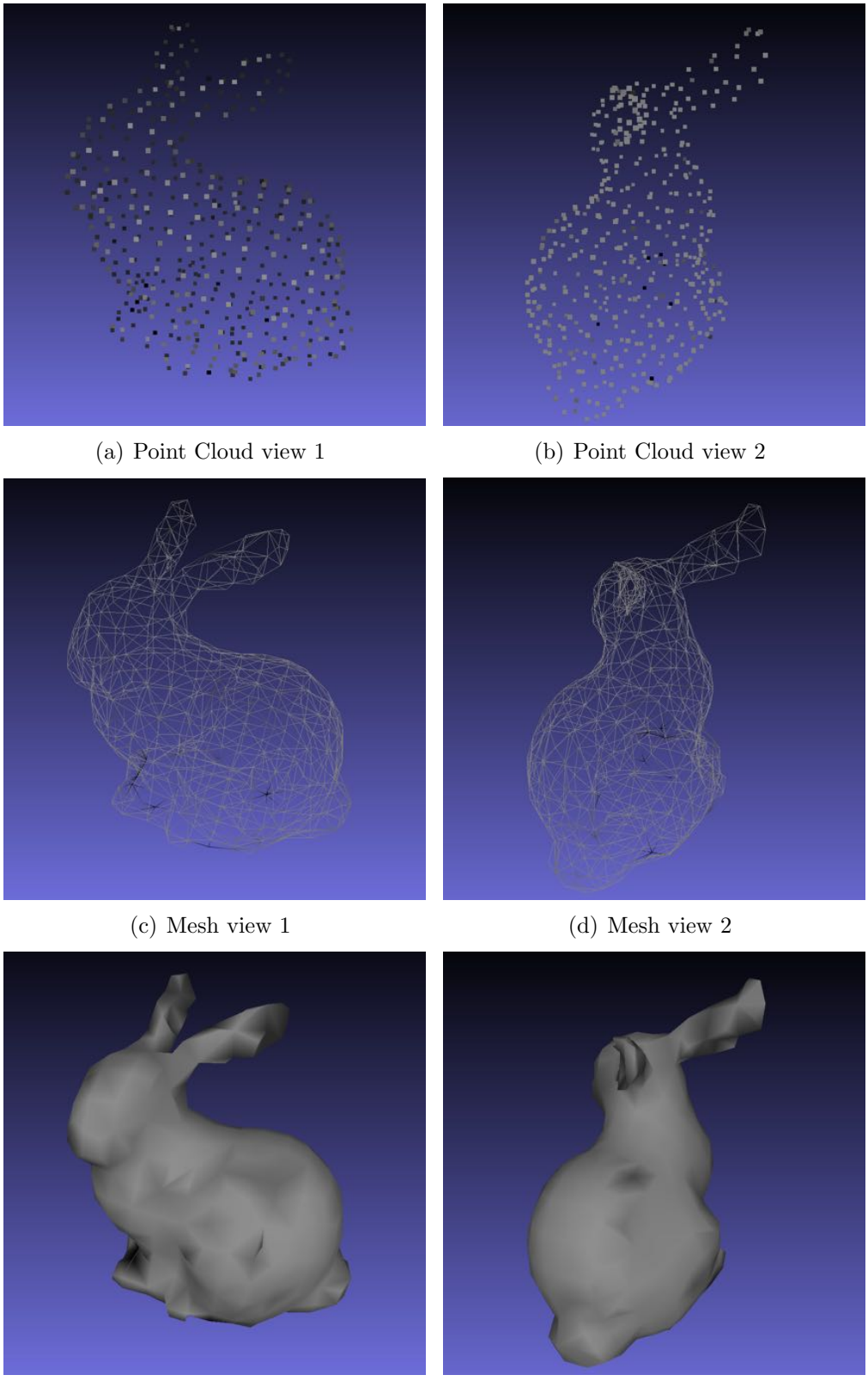


Figure 2.13: Hough transform output of a line. Each point generate one of the curved line. These curve all interset in one point that corresponds to a single line in image space.

A voxel space is a continuous volume a visualizing this type of data is more challenging as every discrete unit of space contains information. It is possible to use maximum intensity projection to have a grasp of what the volume contains but in general visualizing the voxel space requires to give up on some information. The choice between these different representations depends on what information needs to be emphasized.



(a) Point Cloud view 1

(b) Point Cloud view 2

(c) Mesh view 1

(d) Mesh view 2

(e) Surface view 1

(f) Surface view 2

Figure 2.14: Standford bunny, from the Standford 3D dataset collection [31]

Chapter 3

Related Work

This thesis work is closely related to several other approaches and tries to increment upon it. This chapter gives an overview of previously developed methods. It mainly focuses on traditional multi view reconstruction even though a mention of reconstructions from single view is also made. The last sections in this chapter are presenting site modeling and its history, and situating the contribution of this thesis work.

3.1 Bundle Adjustment

Bundle Adjustment is critical preliminary step in many 3D reconstruction pipeline. It is always formulated as an optimization problem that minimize the projection error for a refined set of tie points. For a satisfying result, a sufficient number of tie points need to be used, but the number of parameters to optimize grows proportionally to this number of points and the number of views. The problem can therefore rapidly become extremely complex. This is why this problem has been largely studied and many different approaches were developped[32]. The preferred optimization method is usually the LevenbergMarquardt [33, 34, 35]. Although this thesis focuses on

the dense reconstruction part, from images and refined camera pose obtained from MUB4S by aliakbarpour et al[36, 37, 38] or Visual SFM [39], it is worth mentioning the work done one that critical part without which the desne reconstruction is not possible.

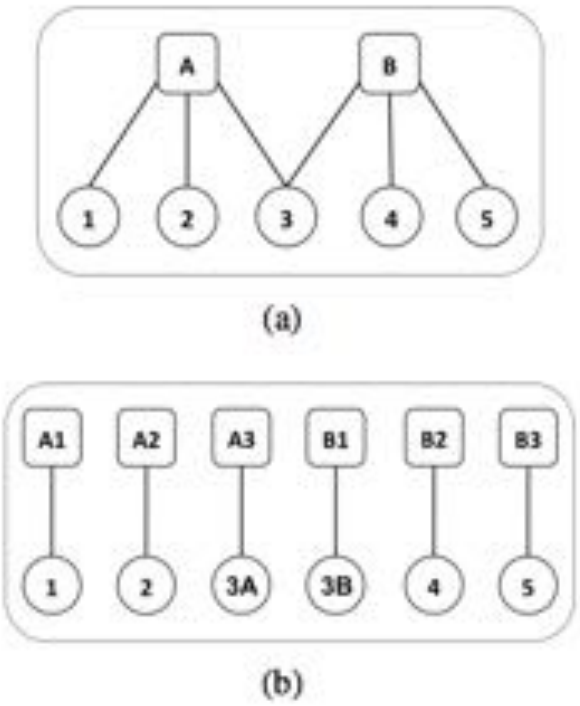


Figure 3.1: A possible approach for bundle adjustment: instead of two cameras looking at a partially overlapping set of points, All the possible camera-point pairs are instantiated and the equivalence between for example cameras A1 to A3 are treated as a constraint of the optimization problem.[40]

3.2 Stereo 3D Reconstruction

3D reconstruction from photo-consistency, in its simplest form starts with stereo reconstruction and computation of a disparity map between two images. Several torough suveys have been done on stereo reconstruction [41, 42]

The search of matching patches between the paired images can be done using

the epipolar constraint by computing the fundamental matrix, or it can also be done using plane sweeping. One of the state of the art method this thesis work is inspired from is the local plane sweeping from Sinha et al [43] (Figure 3.2). After matching features, the use of GPU to rapidly test several orientation gives promising results on standard stereo datasets. Hirschmuller et al using Semi-Global Matching of mutual information also achieved highly accurate results on Stereo reconstruction [44, 45, 46] .

In general, inference from a unique image pair cannot give a full 3D model of the scene which is why the output format is generally kept as a depthmap. This type of technique also require the two views to be close one to another in order to have sufficient overlap and consequently are only a first step toward the obtension of a full 3 model for a real object. Stereo reconstruction method tends to allow less ressource towards handling occlusion for that reason too. The two point ov view being very close together, what is visible on one image is expected to remain as such in the other one.

Multi-view 3D reconstruction can often be framed as a generalization of stereo reconstruction. Comprehensive surveys of multiview algorithms are already available [47, 48], proving that a lot has already been achieved. The rest of this chapter tries to classify the different approaches of interest.

3.3 Patch Based Methods

Also referred to as feature matching based methods. The basic concept is that a feature patch is match through a set of images, and the corresponding 3D point is then triangulated from the 2D images correspondances. If the camera pose has already been estimated through bundle adjustment, the search space can be reduced by matching a point descriptor from the reference image only along the associated

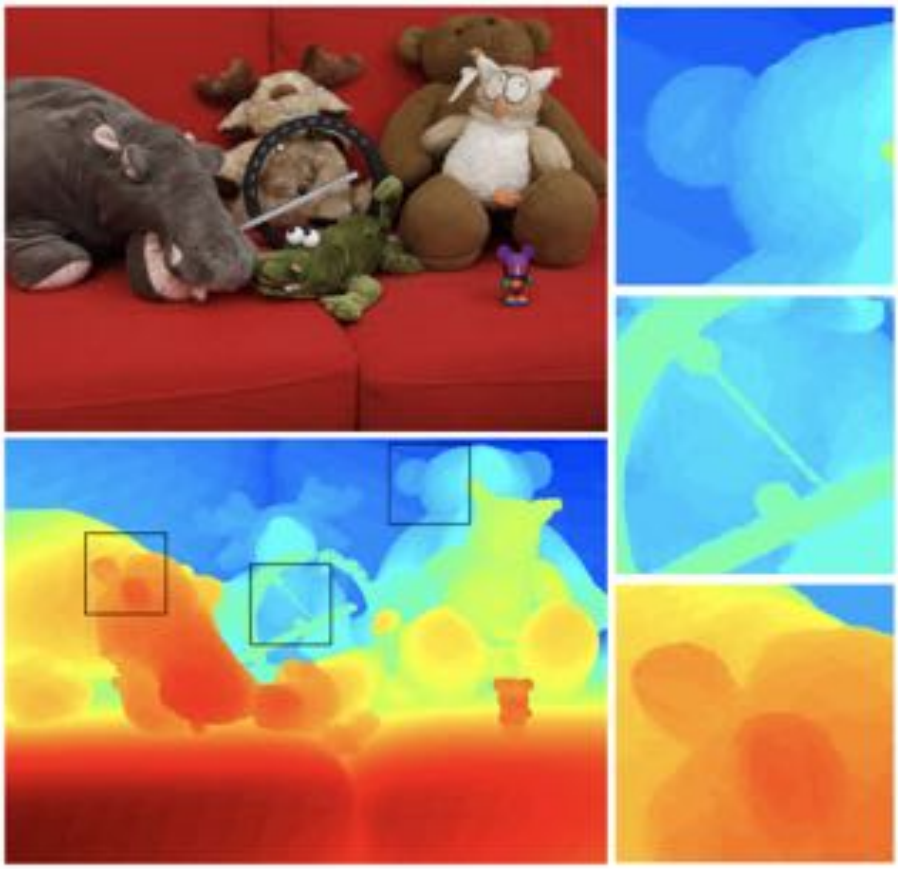


Figure 3.2: Result of Local Plane sweep by Sinha et al [43].

epipolar line in the comparison images. It is still computationally expensive to build a dense 3D model with this approach. EPFL is matching along the epipolar line using DAISY descriptors [49, 50]. One of the top performing algorithm, PMVS [9] also uses this approach, matching SIFT points, and expands surfaces from initally matched patches (Figure 3.3). The search for the nearest patch can also be randomized and give good results as shown by Barnes et al [51].

3.4 Volumetric Methods

The common principle in all volumetric methods is to sample the 3D space and to evaluate for each voxels wether or not the space is occupied. Early methods

Figure 3.3: PMVS reconstruction result from Furukawa et al. [9]

were also referred to as space carving [52, 53, 54]. The basic principle being that a surface voxel would be considered empty if not photoconsistent and the object would progressively take form as if it was carved from a stone. A survey by Slabaugh et al[55] gives a comprehensive overview of early methods through 2001. Many of space carving methods require segmentation of the object as first step [56, 57]. Most recent works instead of binary decision prefer to compute a probabilistic value that can better handle scene uncertainty and ambiguity for novel view generation. This was successfully applied to change detection by Polard et al [58, 59] and then later by Crispell et al[19]. Also as a volumetric approach, Ulusoy at al [60] used Markov Random Field to tackle the scene uncertainty and ambiguity problem.

The principal limitation of the volumetric approach is the memory it requires, the resolution-related issues, and the bounding box localization as this approach typically requires to intialize a bounding volume. The number of voxels grows with the cube of the voxel length, and increasing the resolution becomes quickly memmory prohibive if no optimization are made. The number of operations is also directly related to the number of voxels.

Figure 3.4: An illustration of Novel view generation by Crispell et al. [19]

3.5 Surface Fitting Method

Another type of methods matches point from between images and triangulate them before fitting a surface to it, possibly as a 3D generalization of active contour. This is very unlikely to do well with our type of data, which is why most group dont really apply this sort of methods to WAMI datasets. The clutter in the scene, especially in urban area with alternance of high buildings and streets makes the outcome of such approaches very uncertain. However, Calakli et al [61, 62] use Smoothed Signed Distance to reconstruct a surface either from oriented points cloud or probabilistic volumetric grid, with promising result on aerial imagery.

3.6 Other Notable Work

3.6.1 Single View 3D Reconstruction

Some great work was done in an attempt to extract 3D model from a single view. While most stereo or multi view algorithm are using at one point or another photo consistency constraint, it is in this case not possible. Other clues need to be used such as lightning model in the case of shape from shading [63] or intersection of parallel

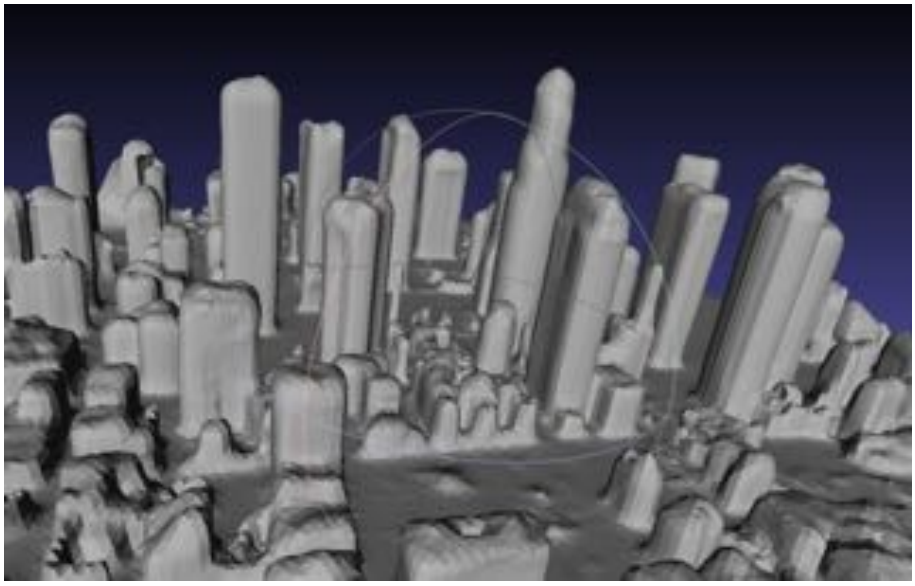


Figure 3.5: SSD from Calakli et al [61] applied to MU3D Los Angeles Point Cloud.

lines at a vanishing point to model a piecewise planar world [64]. These methods are however hardly applicable to Aerial Imagery where the target of the reconstruction is a highly diverse and complex scene.

3.6.2 Deep Neural Network

The available datasets usable to train a deep neural network for 3D reconstruction are still relatively rare although their availaibility is increasing. In general, a learning based approach is not as obvious and straightforward as for a classification task, but the incredible results obtained with CNNs on object classification and even detection obviously raised hope to obtain the same kind of outstanding improvement in 3D reconstruction. Furukawa et al are now working on deep neural networks trained on RGBD data [65]. It is uncleared whether their approach can be applied to Wide Area Imagery. Nonetheless, there are more and more learning based approaches in order to try to learn features that are robust to changes in illumination and point of view [66, 67, 68, 69], and as shown by Paschalidou et al [70], there is potential for learning methods applied to 3D reconstruction from Aerial Imagery.

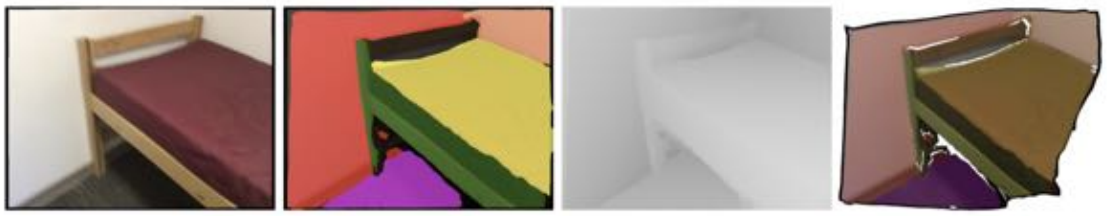


Figure 3.6: An example of output from PlaneNet.[65]

3.7 Site Modeling

Site modeling consists in constructing a 3D model of an geographic area using sensor data, including but not necessarily limited to aerial imagery. Reconstruction of urban scene can also be done with street level imagery or combination of both. This is an obvious potential application of this research work. Most methods were for a long time heavily manual and development of effcient automatic algorithm is of great value for that purpose. Google now has 3D model of main area of biggest city in the US. It can be very important data for mapping, change detection, damage assessment in catastrophic event. This is still a heavily manual process and a lot of effort is made to try to automatize as much as possible what is otherwise very tedious work.

3.8 Thesis Contribution

The main contribution of this thesis work is the combination of several of best performing approaches to tackle the specific challenges raised by our own datasets. The presented work demonstrates how voting with repeatable features can be used to efficiently traingulate a very high number of points without bruteforce descriptor matching in the whole set of images. Different processing steps are presented to either optimize accuracy or completeness.

Figure 3.7: Screenshot of Empire State Building and its surroundings in Google map.

Chapter 4

Voting Pipeline

4.1 Introduction

The camera pose estimation problem is generally solved by minimizing the reprojection error of a relatively reduced number of points that were matched over a subset of images. A typical implementation would detect and match SIFT points before minimizing a cost function. This problem is refered to as Bundle Ad- justment, and although state of the art methods lead to very high precision in the estimate of the cameras position and orientation, the number of triangulated points is too low to render the structure of the scene. Trying to match more points would rapidly increase the computation cost and dense recontruction is therefore not at- tainable this way. Different strategies exist to efficiently render dense structures. For simple objects, space carving coupled with image segmentation techniques can give good results but is not a preferred solution for complex scenes. If match- ing patches between images is still required, once a precise camera pose estimate has been obtained, the epipolar constraint can be used to reduce the search space. The search along the epipolar line, using sum of absolute difference or normalized cross-correlation has proven to be a

robust approach but matching every feature point even after using the epipolar constraint, could still end up being computa- tionally expensive. The proposed approach tries to take advantage of the assumed repeatability of feature points detection over a sequence of images taken from ad- jacent views to avoid the use of an expensive matching function. The following of this chapter is organized as follows:

4.2 Pipeline Description

After data has been acquired from airborne camera and platform sensors such as IMU and GPS, our 3D reconstruction pipeline (Figure 2.2) first refines the cam- era pose estimate with a fast Structure-from-Motion algorithm, and runs a feature point detector on each image in order to select affine stable points that are the most likely to be repeatably detected over several consecutive frames [72, 73]. This choice of feature point is explained by the fact thaqt we need the detection to be repeatable ofr accumulation to occur, but the algorithm could easily be generalized to other feature points such as Harris corners [74] or lines [75]. Estimated camera intrinsic and extrinsic parameters and the sets of feature points are then used by our voting algorithm. After the voting phase is complete, some post process- ing tasks are necessary to separate real structure from inherent voting noise. Then, the 3D structure extraction can be performed by either automatically or manually thresholding the processed vote space. Finaly, only one last step remains, in order to obtain photo realistic rendering, the raw uncolored point cloud is reprojected on a subset of images in order to assign a color to each point.

This thesis work focuses on the dense reconstruction part that starts after bundle adjustment (Figure 4.2).

The following of this section will consecutively describe into further details:

- Voting

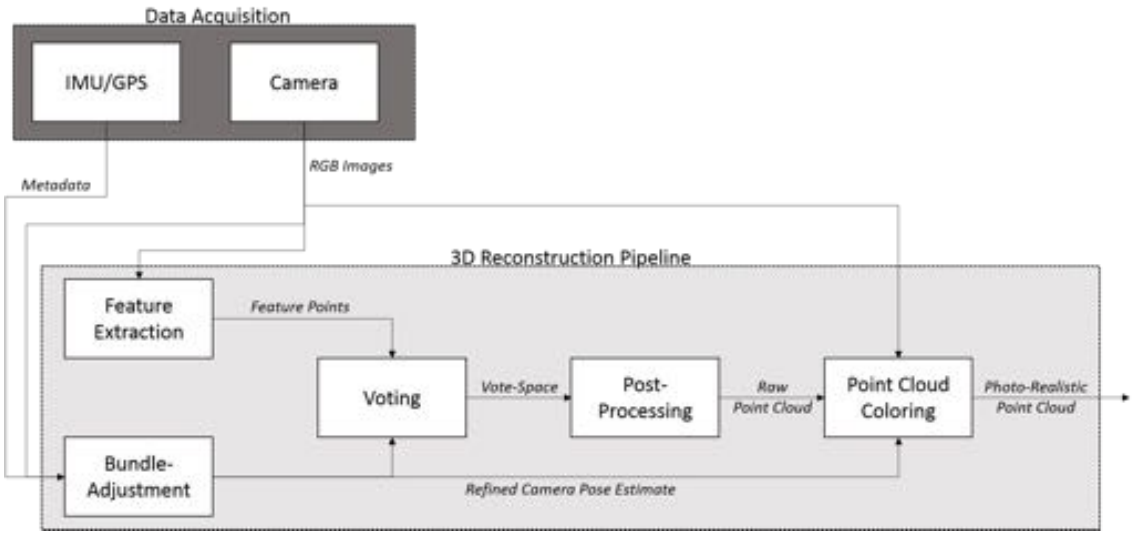


Figure 4.1: Feature Based Voting Pipeline: From RGB high resolution images and metadata to dense photo realistic point cloud.

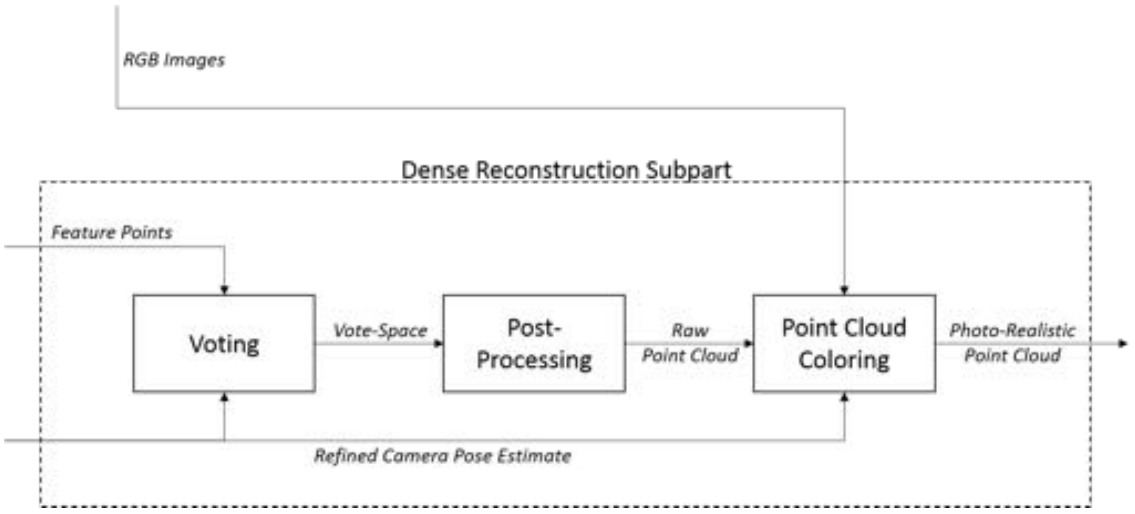


Figure 4.2: Dense Reconstruction part: from refined camera pose estimates and sets of feature points to photo-realistic point cloud.

- Post-Processing
- Point Cloud Coloring

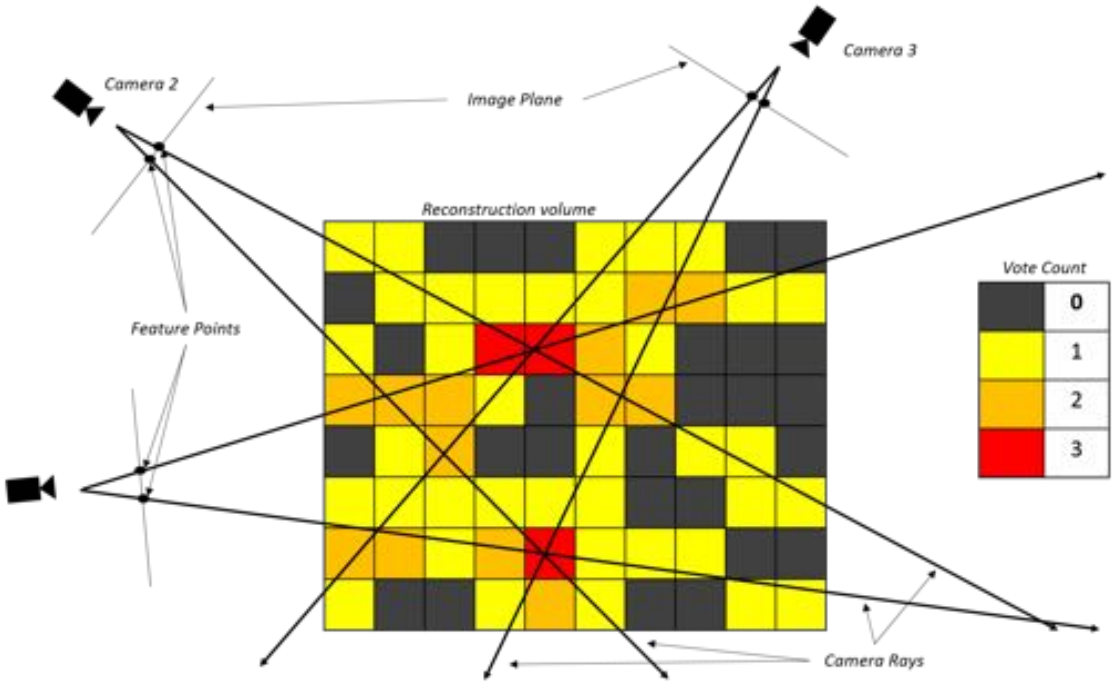


Figure 4.3: An object with two corners (or point of interest) creates for 3 different point of view, images each with 2 detected feature points. Without explicitly matching them and only by casting rays through a voxel box, the location of the two points of interest can be infered from the vote accumulation.

4.2.1 Voting

First part of the dense reconstruction, voting is a volumetric approach. A Voxel box is computed around the targeted area and for each feature point of each image, a ray is computed. We then cast these rays from the camera center of projection, through the Voxel box, and increment the vote count of a voxel as many times a ray is traversing it (Figure 4.3). The algorithm can be summarized as follows: Given a set of images and an accurate estimate of the camera positions, the orientation and the optical parameters:

The method relies on the consistent and repeated detection of the same feature point over several consecutive images. Detection are more likely to be repeated over views that are very close in time. However, projections of a same point on two very close views give almost identical rays, and would reduce the precision of the

Algorithm 1 Voting Algorithm

```
1: VoxelBox = InitializeVoxelBox
2: for ImageIndex = 1 : NImage do
3:     FeaturePointList = GetFeaturePointList(ImageIndex)
4:     CameraParameters = GetCameraParameters(ImageIndex)
5:     for FeatureIndex = 1 : NFeatures do
6:         Ray = GetRay(FeaturePointList(FeatureIndex), CameraParameters)
7:         [EntryPoint, ExitPoint] = GetRayAndVoxelBoxIntersections(Ray, VoxelBox)
8:         VoxelList = Bresenham3D(EntryPoint, ExitPoint)
9:         for VoxelIndex = 1 : NVoxel do
10:            VoxelList(VoxelIndex) + +
11:        end for
12:    end for
13: end for
```

triangulation (See Figure 4.4). Ideally, we would want an angle as close as possible to 90 degrees for an optimal triangulation precision (Figure 4.5), but it is less likely that a a point on a scene would consistenly trigger a corner detector on images separated by that angle.

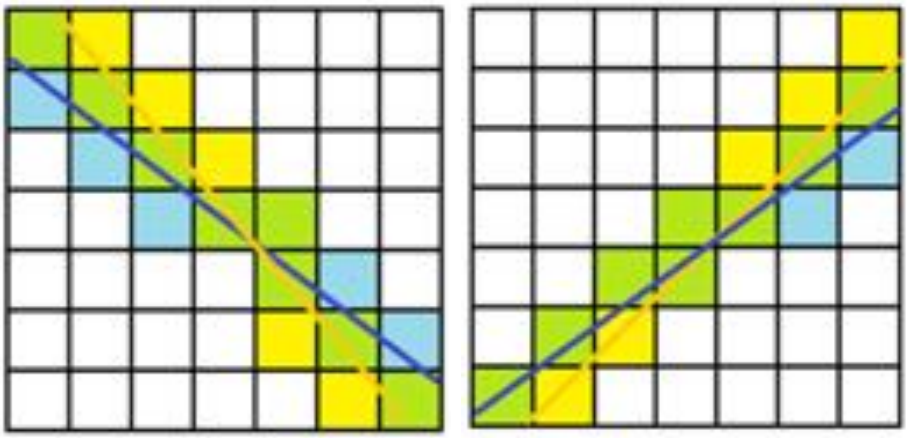


Figure 4.4: Two examples of two consecutive rays going through a same point. Pixels traversed only by yellow (resp blue) ray are colored in yellow (resp blue), pixels traversed by both rays are colored in green. As they are closed in time, the angle between each pair of rays is small and they therefore share a lot of pixel, preventing from locating with precision the intersection point. Obviously, the number of pixel in common will also depend on the resolution

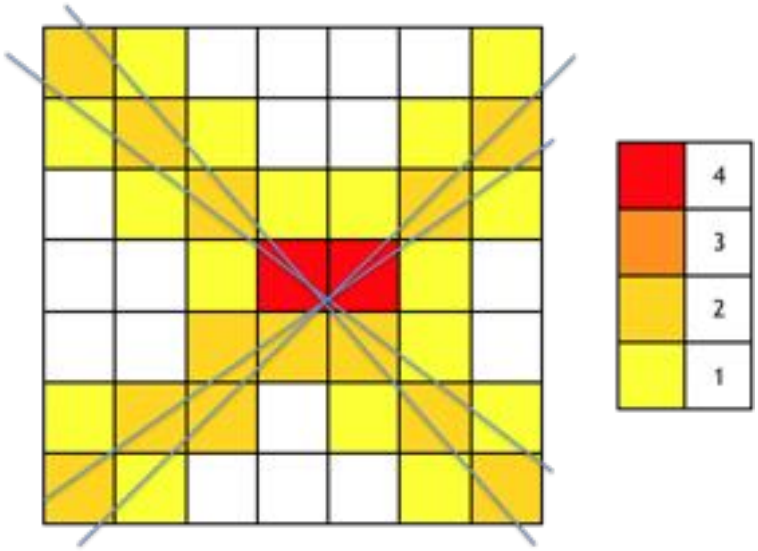


Figure 4.5: If we superimpose the contribution of the four rays of figure 2, the number of votes for each pixel ranges from 0 to 4. Using rays with a bigger angle between them, therefore more distant in time allowed getting a more accurate estimate of the position

4.3 Post-Processing

Right after voting, the main post processing phase is implemented in four steps:

- Vote Collapsing
- Gradient Computing
- Horizontal smoothing
- Automatic or Manual Thresholding

4.3.1 Vote Collapsing

The Vote collapsing algorithm is a first way to increase the density of 3D points in the final point cloud. It is based on the assumption that detected structures cannot float in the air and should be supported underneath by a base. Therefore, as the number

of votes is evidence of occupancy for any given voxel, no voxel should have less votes than the voxels directly above it. This constraint is enforced using the algorithm 2:

Algorithm 2 Vote Collapsing

```
1: for x=1:Xmax do
2:     for y=1:Ymax do
3:         for z=Zmax-1:1:-1 do
4:             Voxel(x, y, z) = max(Voxel(x, y, z), Voxel(x, y, z + 1))
5:         end for
6:     end for
7: end for
```

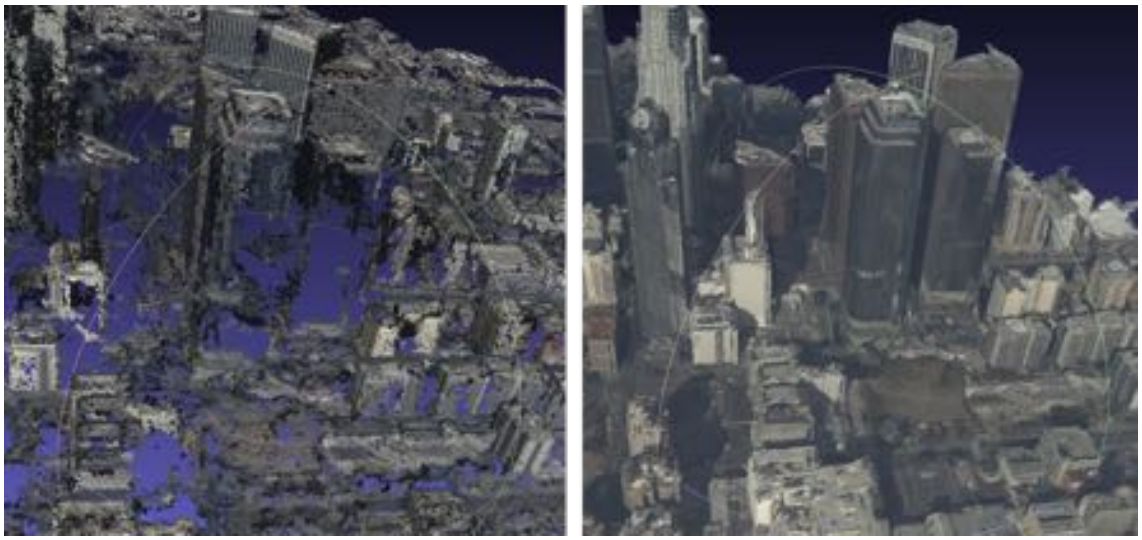


Figure 4.6: Left: Without vote collapsing, right with vote collapsing. Vote collapsing allows to reconstruct homogeneous building facades that would otherwise not be infered with enough confidence if only using photo-consistency.

4.3.2 Gradient Magnitude

Experience has proven that we obtain better results by thresholding using the gradient magnitude instead of the number of votes. The gradient magnitude computation described in algorithm 3, is straight forward.

Algorithm 3 Gradient Magnitude

1: **for** x=2:Xmax-1 **do**
2: **for** y=2:Ymax-1 **do**
3: **for** z=2:Zmax-1 **do**
4: $GradientMagnitude(x, y, z) = sqrt((Voxel(x + 1, y, z) - Voxel(x - 1, y, z))^2 + (Voxel(x, y + 1, z) - Voxel(x, y - 1, z))^2 + (Voxel(x, y, z + 1) - Voxel(x, y, z - 1))^2)$
5: **end for**
6: **end for**
7: **end for**

4.3.3 Horizontal Smoothing

A smoothing step is implemented on the Gradient magnitude, using a gausian Kernel. For efficienty, we loop on each sampled altitude and perform a 2D convolution (See algorithm 4):

Algorithm 4 Horizontal Smoothing

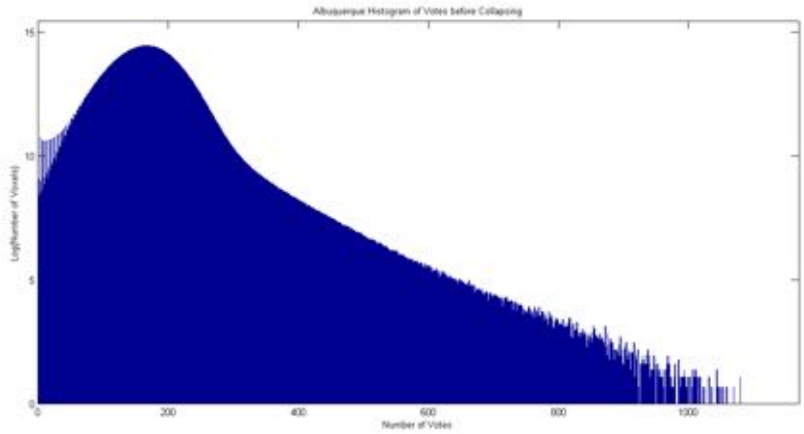
1: $K = GaussianKernel(Size, Sigma)$
2: **for** z=1:Zmax **do**
3: $GradientMagnitude(:, :, z) = K * GradientMagnitude(:, :, z)$
4: **end for**

4.3.4 Histograms Computing

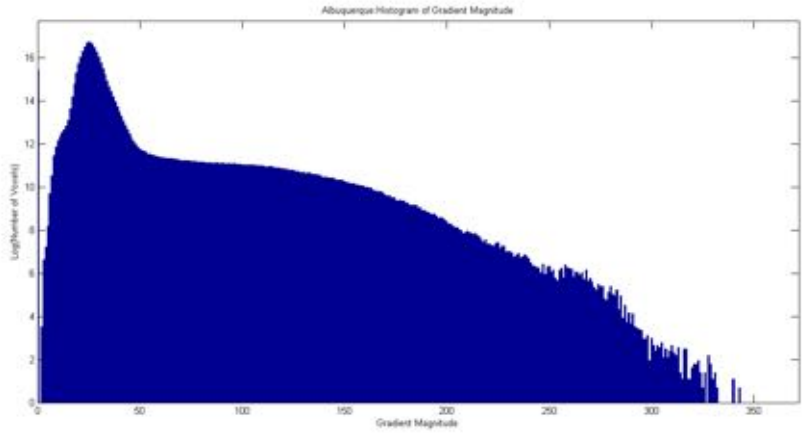
Depending on the tresholding method that comes next, this phase can be necessary or optional. Either way it is straight forward, We simply compute the histograms (Simple and Cumulative) of values for both Votes and GradientMagnitude. The size of the bins is set to one. (See example in Figure)

4.3.5 Combination of Information for Automatic Thresholding

The number of votes, the range of the gradient magnitude depend on numerous parameters as for example the number of frames used for voting and the grid resolution. In case of automatic thresholding it becomes necessary to normalized the information contained in the voxel box. Therefore, we use the histograms to define a low threhold



(a) *Histogram of votes*



(b) *Histogram of gradient magnitude*

Figure 4.7: Example of histograms for Albuquerque Dataset

below which the voxel score is set to 0 and a high threshold above which the score is set to 1 and use a sigmoid function (Figure 4.8 and equation) to assign a normalized score between 0 and 1 to any value between the low and the high thresholds. The standard equation for a sigmoid function is:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

In this particular case, we have $S(0) = 0.5$ and S tends to 0 for negative values of x, and to 1 for positive values. This can be adjusted by composing with an affine

transformation so that the transition from 0 to 1 is delayed and slower.

The low threshold is chosen as the mean value, and the high threshold as the value such as only 2 percent of the voxels are above it. Two independant scores are computed, one for the number of votes and one for the gradient magnitude and for each voxel the two scores are then multiplied together to obtain a final score normalized between 0 and 1. Usually we keep in the final point cloud only voxel with a value above 0.5.

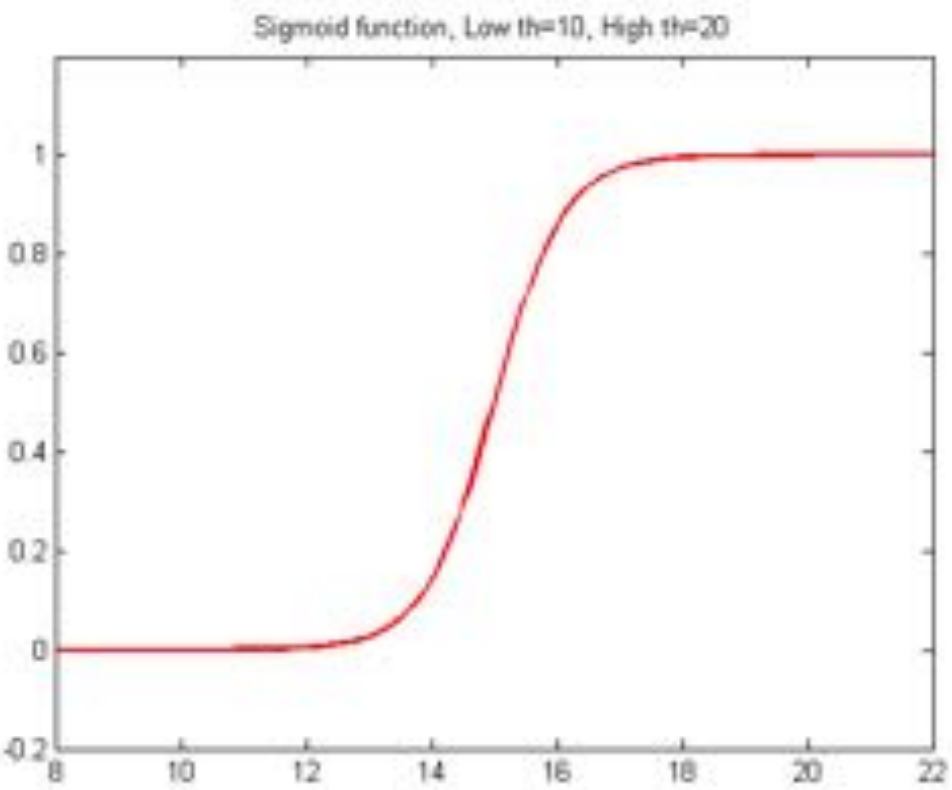
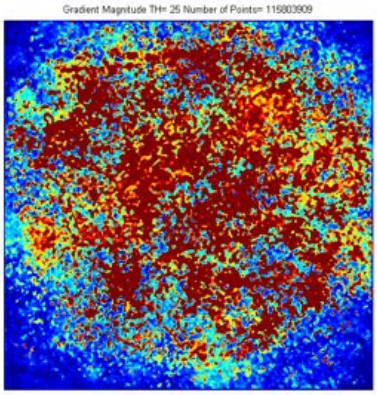


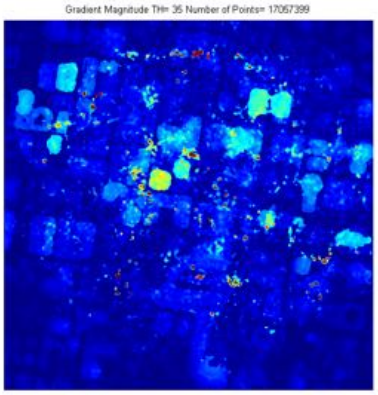
Figure 4.8: Example of sigmoid used as a transition function, the low threshold is set to 10 and anything below it is given the probability 0, any value above 20 is set to a probability of 1 and we have a continuous transition between the two thresholds.

4.3.6 Manual Thresholding

Alternatively is is also possible to manually threshold the volume. In interest of time, we avoid generating a point cloud every time a threshold value is tested. Instead we quickly compute a 2D image showing a top view of the expected point cloud for a serie of values until we narrow down a suitable threshold (see figure 4.9).



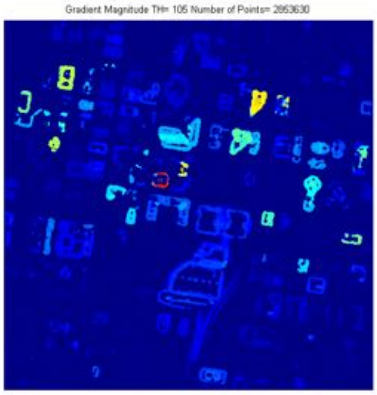
(a) *TH=25*



(b) *TH=35*

(c) *TH=55*

(d) *TH=75*



(e) *TH=95*

(f) *TH=115*

Figure 4.9: Example of outputs of the vizualizer used to threshold manually for Albuquerque

4.4 Point Cloud Coloring

The previous steps only allow to extract the structure but not the appearance. Hence, once the volume has been thresholded, each point still needs to have a color assigned

to it. Besides, a significant amount of the points present in the raw point cloud are actually interior point that are not visible on the images. The proposed coloring algorithms deal with occlusion when assigning a color to each point and eventually remove from the point set, any point that would never be visible on the tested views. There are actually two possible algorithms:

- One Algorithm is Bresenham based, and strictly enforces occlusion but is time expensive
- One Algorithm is faster but has a lower accuracy with regard to the occlusion constraint

4.4.1 Bresenham Based Coloring Algorithm

Given the initial number of points and the computation cost to determine the visibility, we cannot afford to run the algorithm on too many views and have to select a subset of the images used for the reconstruction. Usually this subset contains around 10 views. For each image of the subset, we go through every point of the raw point cloud and check for its visibility on the current image using bresenham to determine if another occupied voxel stands in between this point and the cam- era center. If the point is visible, after back projecting it on the image, the color read on the image is assigned to it. Once a color is assigned to a point it cannot be modified and we will not even check for its visibility in the next images. This con- straint is to avoid mixing different views to color adjacent points. the algorithm can be summarized as follows:

4.4.2 Z-Buffer Based Coloring Algorithm

This alternative algorithm has the obvious advantage to be faster. The Bresenham algorithm is computationaly expensive. Projecting every point once is still obvi- ously

Algorithm 5 Bresenham based coloring algorithm

```
1: ImageSubset = ExctractImageSubset()
2: for ImageIndex = 1 : NImagesInSubset do
3:    for PointIndex = 1 : NPoints do
4:       if IsColored(PointIndex) == false then
5:          VoxelList = Bresenham3D(CurrentPoint, CameraCenter, VoxelBox)
6:          if AllVoxelsAreEmpty(VoxelList) == true then
7:             ImageProjection = GetImageProjection(ImageIndex, PointIndex)
8:             Color(PointIndex) = ImageProjection.Color
9:             IsColored(PointIndex) = true
10:         end if
11:      end if
12:   end for
13: end for
```

a necessity, but instead of searching all along the ray for other occluding voxel it is more efficient to keep track for each location on the projection plane of the projected point with minimal depth. After every point has been projected, the list of visible point can easily be retrieved and updated with the appearance infor- mation from the current image. A cost function such as the local depth or depth gradient can be used to decide wether or not to update. However, since points have no dimension, the well enforcement of the occlusion constraint is dependant on the depthmap resolution that need to be lower than the one of the point cloud.

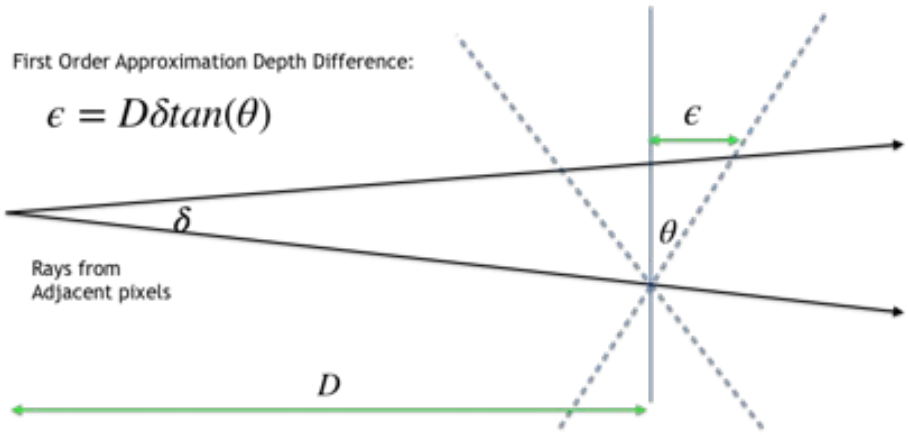


Figure 4.10: A larger angle between the ray and the surface create a larger depth difference between adjacent pixels

Algorithm 6 Z-Buffer based coloring algorithm

```
1: ImageSubset = ExctractImageSubset()
2: DepthMap=assign < float > (DepthMapWidth, DepthMapHeight)
3: IndexMap=assign < int > (DepthMapWidth, DepthMapHeight)
4: Cost=assign < float > (NPoints)
5: Cost.fill(INF)
6: for ImageIndex = 1 : NImagesInSubset do
7:    DepthMap.fill(INF)
8:    IndexMap.fill(−1)
9:    CurrentImage = LoadImage(ImageIndex)
10:   for PointIndex = 1 : NPoints do
11:      [px py Depth]=BackProject(PointIndex, ImageIndex)
12:      [x y]=Image2DepthMap(px, py)
13:      if Depth < DepthMap(x, y) then
14:         DepthMap(x, y) = Depth
15:         IndexMap(x, y) = PointIndex
16:      end if
17:   end for
18:   for x = 1 : DepthMapWidth do
19:      for y = 1 : DepthMapHeight do
20:         if IndexMap(x, y)! = −1 then
21:            CurrentPointIndex=IndexMap(x, y)
22:            CurrentCost=CostFunction(x, y)
23:            [px py]=DepthMap2Image(x, y)
24:            if CurrentCost < Cost(CurrentPointIndex) then
25:               Cost(CurrentPointIndex) = CurrentCost
26:               Color(CurrentPointIndex) = CurrentImage(x, y).Color
27:            end if
28:         end if
29:      end for
30:   end for
31: end for
```

4.4.3 Complexity Analysis

The fast Z-Buffer algorithm presents several advantages compared to its Bresenham based counterpart. In term of memory complexity, it onlyh requires to maintain a list of points, a depthmap and an image, whereas the Bresenham based algorithm requires to keep a 3D grid in memory. The depthmap resolution is tied to the point cloud resolution in such a way that it is always lower than the input image resolution. We can therefore consider that the memory complexity of Bresenham based and Z-buffer based coloring algorithms are respectively $\mathbf{O}(N_{voxels} + N_{imagepixels})$ and $\mathbf{O}(N_{points} + N_{imagepixels})$. As most of the 3D grid is usually empty, the memory usage is divided

by at least a factor of 10. In term of execution time, the Z-buffer algorithm scales linearly with the number of points and the number of views ($\mathbf{O}(N_{views} * N_{points})$ while the brensenham based algorithm is dependant on the number of voxels (($\mathbf{O}(N_{views} * N_{voxels})$)

4.5 Experimental Results

4.5.1 Four Hills

Four Hills dataset sheds light on the limits of the approach. The high variation of elevation in general makes the approach fall apart. Since this dataset has lower frame rate and lower image resolution, less time has been spent optimizing the result, as the new frame rate and resolution was going to be the new baseline moving forward.

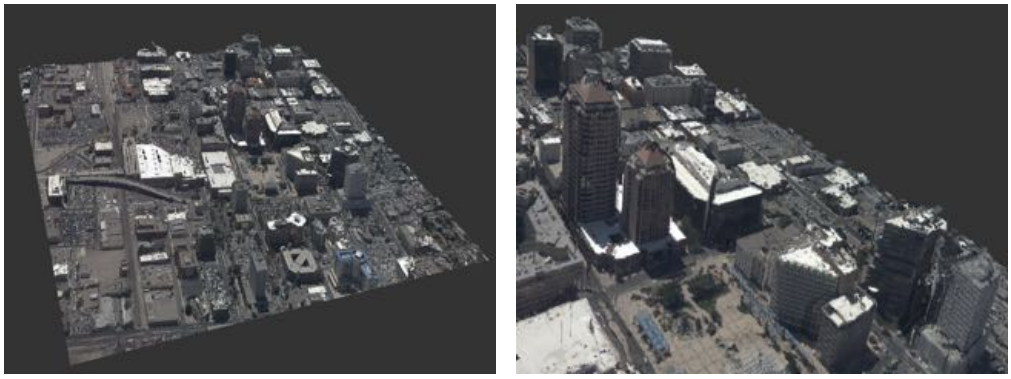
(a) *Four Hills overall view* (b) *Zoom in bridge and residential area*

Figure 4.11: Screenshots of the vizualized output for Four Hills

4.5.2 Albuquerque

The presented approach works very well on Albuquerque dataset. A lot of the structure is correctly reconstructed, notably around the downtown plaza. Thje result have a few obvious flwas, like the large white roof of the convention center that cannot

be completely reconstructed, unless we use an additional post processing step with morphology. However this additional processing has its downside as it tends to merge together buildings that are to close from one another.

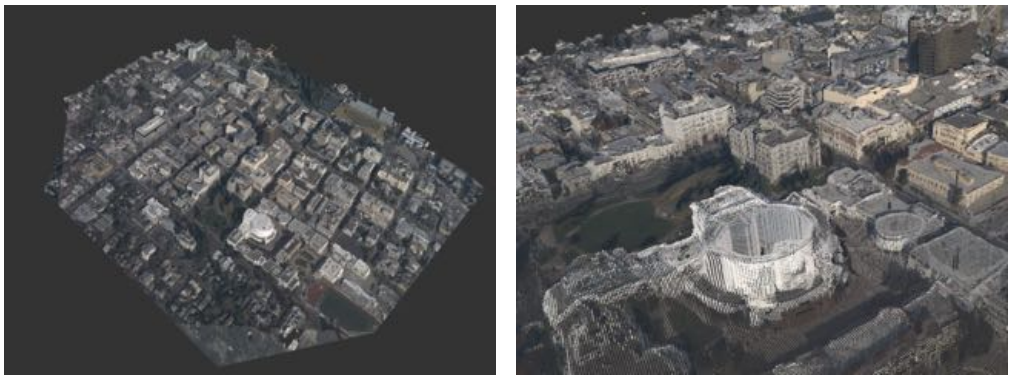


(a) *Albuquerque overall view* (b) *Zoom in Downtown*

Figure 4.12: Screenshots of the vizualized output for Albuquerque

4.5.3 Berkeley

Berkeley, is another example of dataset on which the voting pipeline is relatively successful. The moderate size buildings are reconstructed with good accuracy. Some building with homogenous rooftops have holes and the stadium area already shed lights on some of the pipeline limitations.

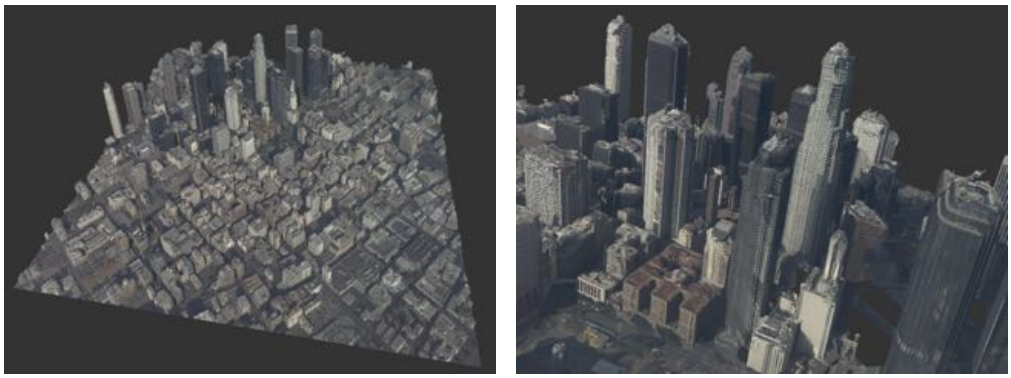


(a) *Berkeley overall view* (b) *Zoom in Downtown*

Figure 4.13: Screenshots of the vizualized output for Berkeley

4.5.4 Los Angeles

Los Angeles provides the most satisfying result. The algorithm seems to perform particularly well with this type of dataset. All the tall buildings in the downtown area are reconsgtructed with high fidelity. The relatively flatter area is also rendered with a very good quality. To name a few rare downsides, the extrusion or vote collapsing causes a crane to turn into a wall and the lack of usable views of the skyscrapers basis causes the presence of holes or shadows in that area.

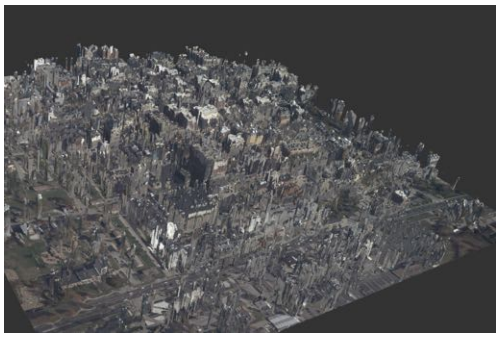


(a) *Los Angeles overall view* (b) *Zoom in Downtown*

Figure 4.14: Screenshots of the vizualized output for Los Angeles

4.5.5 Columbia

The original voting pipeline does not perform well on the Columbia dataset. The elevation variation poses a challenge not properly handled by this pipeline, and this dataset makes that flaw very obvious. Depending on where the bottom plane is set, we either get empty areas or a "house on pillar" effect. Another possible explanation is that the feature detector also happens to under perform on that specific image sequence. The result indeed shows very sparse and discontinuous structures.



(a) *Columbia overall view*

(b) *Zoom in Downtown*

Figure 4.15: Screenshots of the vizualized output for Columbia

Dataset	**Voxel Resolution**	**Number of points in output point cloud**
Four Hills	1001x1001x66 (1m)	1,141 Millions
Albuquerque	1001x1001x201 (1m)	1,521 Millions
Berkeley	1001x1001x151 (1m)	1,756 Millions
Los Angeles	1501x1501x401 (1m)	4,727 Millions
Columbia MO	1001x1001x161 (1m)	2,795 Millions

Table 4.1: Summary of Dataset resolution and output point cloud sizes

4.5.6 Scaling Summary

The table 4.1 summarizes the size of each reconstructed dataset:

4.6 Other Experiments

We tried several other experiments to extract the 3D structure from the voting space as rigourously as possible. The rest of this section presents some of the approaches we tried.

4.6.1 Temporal Incremental Voting and Removal

Voting all along a ray, from its Voxel Box entry to its exits, partly explain the significant amount of noise in the final vote volume. This noise increases for each new view and one possible solution would be to remove some of the votes classified as noise at the same time we are adding votes from another new frame. Therefore, we propose a modified voting algorithm that enforces an implicit temporal consistency of votes and filters the noise from older frames at the same time it adds new frame, by alternating phasis of voting and vote removal (See algorithm 9). Besides, this method also enforces that each voxel can received at the most one vote per frame, regardless of the voxel resolution and features density (See algorithms 7 and 8).

Algorithm 7 Voting_Unique(CurrentImageIndex)

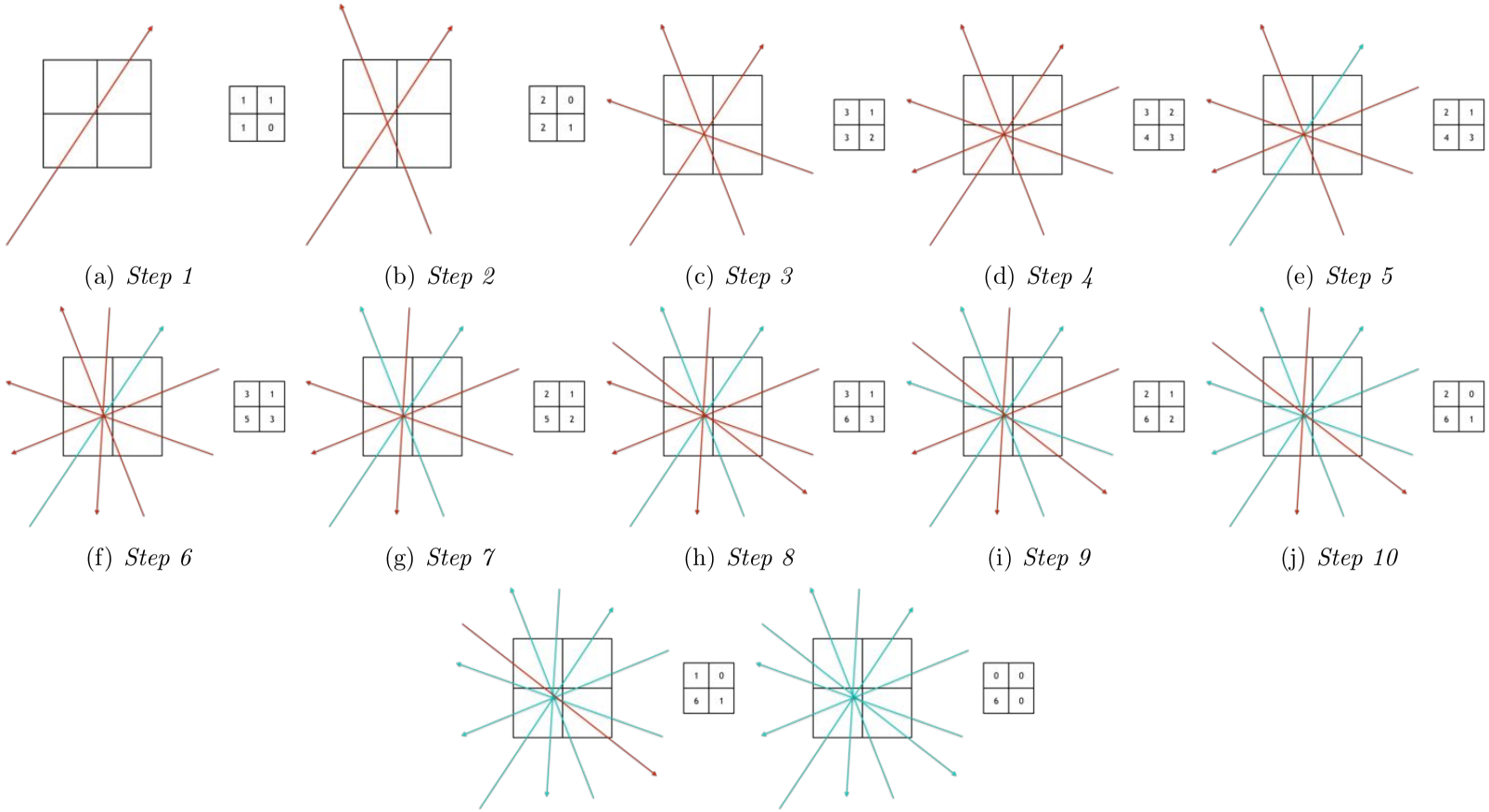
```
1: for All Voxel do
2:    Voxel.CurrentVote = false
3: end for
4: FeaturePointList = GetFeaturePointList(CurrentImageIndex)
5: CameraParameters = GetCameraParameters(CurrentImageIndex)
6: for each FeaturePoint in FeaturePointList do
7:    Ray = GetRay(FeaturePoint, CameraParameters)
8:    [EntryPoint, ExitPoint] = GetRayAndVoxelBoxIntersections(Ray, VoxelBox)
9:    RayVoxelList = Bresenham3D(EntryPoint, ExitPoint)
10:   for each Voxel in RayVoxelList do
11:      Voxel.CurrentVote = true
12:   end for
13: end for
14: for All Voxel do
15:   if Voxel.CurrentVote == true then
16:      Voxel.VoteCount++
17:   end if
18: end for
```

Algorithm 8 Voting_Removal(CurrentImageIndex)

```
for All Voxel do
    Voxel.CurrentVote = false
end for
FeaturePointList = GetFeaturePointList(CurrentImageIndex)
CameraParameters = GetCameraParameters(CurrentImageIndex)
for each FeaturePoint in FeaturePointList do
    Ray = GetRay(FeaturePoint, CameraParameters)
    [EntryPoint, ExitPoint] = GetRayAndVoxelBoxIntersections(Ray, VoxelBox)
    RayVoxelList = Bresenham3D(EntryPoint, ExitPoint)
    for each Voxel in RayVoxelList do
        if Voxel.VoteCount < RepeatabilityThreshold then
            Voxel.CurrentVote = true
        end if
    end for
end for
for All Voxel do
    if Voxel.CurrentVote == true then
        Voxel.VoteCount − −
    end if
end for
```

Algorithm 9 Alternated Unique Voting and Vote Removal

```
for FrameIndex=1 to RepeatabilityThreshold do
    Voting_Unique(FrameIndex)
end for
for FrameIndex=RepeatabilityThreshold + 1 to N_Frames do
    Voting_Removal(FrameIndex − RepeatabilityThreshold)
    Voting_Unique(FrameIndex)
end for
```



(a) *Step 1* (b) *Step 2* (c) *Step 3* (d) *Step 4* (e) *Step 5*

(f) *Step 6* (g) *Step 7* (h) *Step 8* (i) *Step 9* (j) *Step 10*

(k) *Step 11* (l) *Step 12*

Figure 4.16: The algorithm starts by casting votes from 4 consecutive frames (rays in red), and then alternates removing vote from oldest view (rays become blue) and adding votes from new views, until there are no new view left. Only voxels with a vote count below a threshold are decremented.

4.6.2 Occlusion and Appearance Consistency Filtering

deriving depth probability distribution from vote volume

Instead of applying a global threshold, the vote volume can be converted into a probability of occupancy for each voxel, from which can be derived a depth distribution along each ray (See figure 4.17). The order in which the voxels are met is important and the first voxel with very high occupancy probability should correspond to the maximum of likelihood for the depth value, even if some peaks with higher vote value can be met later down the ray. It is actually simple to formalize it. A voxel is visible from a position if and only if all the voxels in between are not occupied. Therefore, given the probability of occupancy for each voxel in between, we can compute the probability of visibility as the product of all preceding voxel probabilities of non-occupancy (See equation 4.2). Then, the current voxel corresponds to the feature observed on the image if and only if that voxel is visible, and occupied. We consider in that case that the depth of the ray is the distance from the camera center to the current voxel, and to simplify without loss of information, we write Depth=i where i is the voxel index, varying from 1 which is the closest to the camera to N which is the furthest apart. (See equation 4.3). Before all of this, the probability of occupancy is computed by normalizing the vote volume, using a sigmoid (see figure 4.8) and two boundaries corresponding to a low threshold, taken as the average number of vote, below which the probabiilty equals 0 and a high threshold that coresponds to the top 2 percent number of votes, above which the probability is set to 1.

$$P_{Visibility}(i) = \prod_{j<i}(1 - P_{Occupancy}(j)) \tag{4.2}$$

$$P(Depth = i) = P_{Visibility}(i) * P_{Occupancy}(i) \tag{4.3}$$

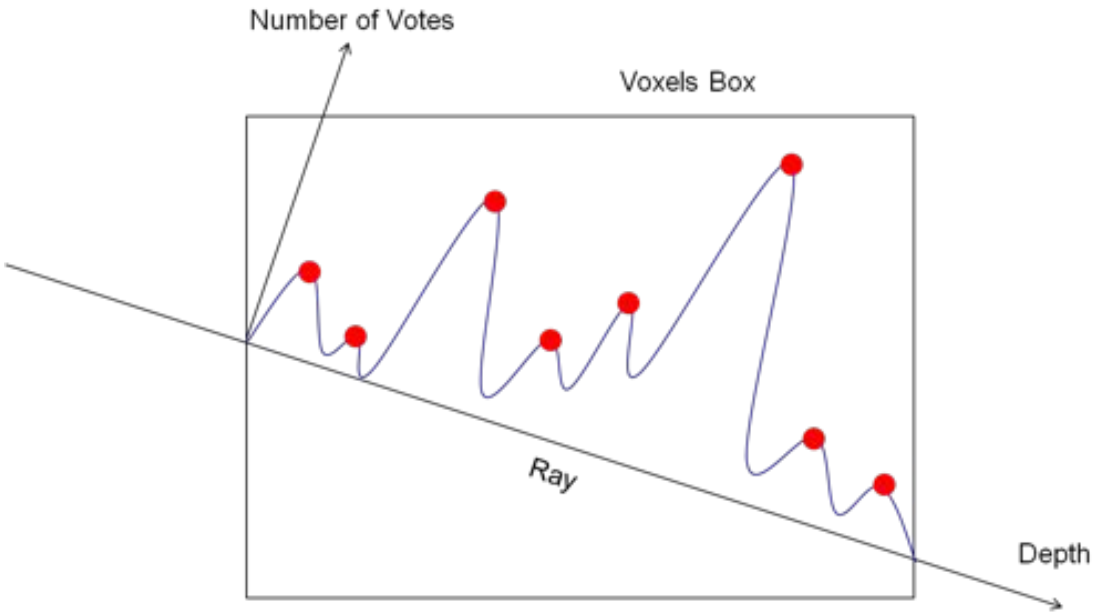


Figure 4.17: The ray goes through the whole voxel box and comes across voxels with various number of Votes. The higher the number of votes, the higher the probability for the voxel to be occupied. The order in which the voxels are met matters and the highest probability of occupancy is in general not the same as the highest probability of Depth. The candidate locations for the real point are taken at each local maximum peak (red dots)

combining with appearance consistency

It is possible to make the model more robust by adding appearance information. Each voxel can be projected on several images, and a matching score can then be computed (using either cross correlation or a descriptor like for example Daisy). If the appearance is not consistent between different views, it should decrease the probability for that voxel to be occupied. However, we do not want the consistency information to take over the depth probability we previously computed. The appearance matching score is normalized into an α value between 0 and 1. A value of 0 should correspond to a total inconsistency and should raise the probability of non-occupancy or transparency to 1. A high consistency should decrease the probability of transparency, but the latter should still depend on the voting information. We therefore propose to use

α as an exponent in the following way:

$$P_{Visibility}(i) = \prod_{j<i}(1 - P_{Occupancy}(j))^{\alpha_j} \tag{4.4}$$

$$P(Depth = i) = P_{Visibility}(i) * [1 - (1 - P_{Occupancy}(i))^{\alpha_i}] \tag{4.5}$$

Equation 4.5 can also be put in the simpler form:

$$P(Depth = i) = P_{Visibility}(i) - P_{Visibility}(i + 1) \tag{4.6}$$

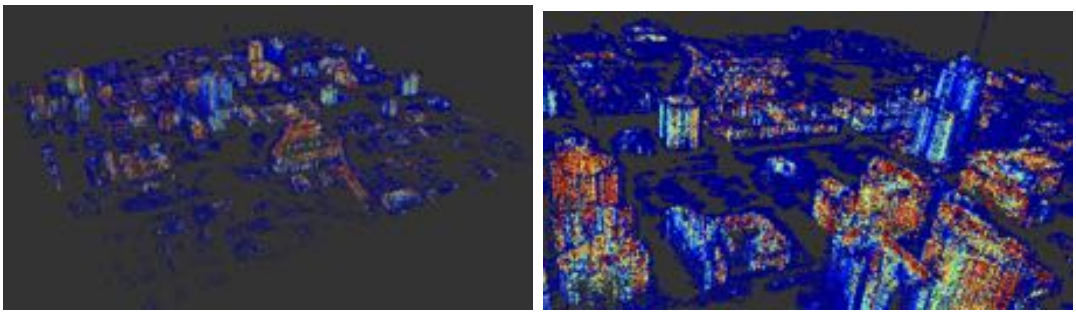


Figure 4.18: A Jet colormap is applied to the point cloud to represent the final confidence. Blue is the lowest confidence, red the highest. As we can see most of the point have low confidence

4.6.3 Thresholding by Vote Accumulation

Another possibility is to keep track of the number of rays computed from feature points and to select the minimal set of voxels such as the sum of all their votes equals this number of rays. This global threshold is always perfectly defined and naturally selects the voxels with higher vote accumulation.

Figure 4.19: Screenshot of a point cloud of Albuquerque Dataset obtained with the vote accumulation thresholding method.

4.6.4 Algorithms Memory Requirement

As a volumetric approach, the voting memory requirement is dependant on the number of voxels ($\mathbf{O}(N_{Voxels})$), which is determined by the size of the bounding volume and the space sampling resolution.

4.7 Discussion: Shortcoming and Limitations

The quality of the result significantly varies from one dataset to another. A scene like downtown Los Angeles, or to a certain extent Albuquerque will allow the method to produce good results but scenes wthl smaller structures are in often only partially reconstructed. There is in general only little garantee over the qual- ity of the final result due to several reasons. First of all, after every feature point from every image has been used to cast a ray, the resulting vote space contains a lot of noise and finding a thresholding method based on strong theoritical ground and leading to a satisfactory result has proven to be difficult. The best results were obtained with ad hoc method and manual intervention. But even with an ideal thresholding method, the final number of true 3D point should be bounded to the average number of feature points per frame and as a result the reconstruction is often only partial. This drawback is however inherent to the method and if the number of feature point per frame is increased, firstly this means using less reli- able features and secondly the amount of noise can rapidly become overwhelming to a point it becomes impossible to extract any structure. It is therefore preferable to be selective when voting and then apply post processing techniques on the vote space such as vote collapsing, gaussian smoothing and morphology. The vote collapsing works as an extrusion and allows to reconstruct most of the building facades and adds a lot of structure as it is very common that only feature points on rooftops would get detected. However, the lack of information on the local terrain often causes to extrude beyond the ground level. In addition, some of the structures like bridges, arches or cranes are non convex and cannot be correctly re- constructed with this method. Similarly Gaussian smoothing was used to derive a continuous density value from the sparse voting information, but while it allows to obtain more continuous structures, this has a cost with regard to accuracy. Then, morphology can be used in ad hoc way to fill holes in wide rooftops that tends to lack textures but it can sometime cause unwanted merges

between neighboring buildings. Last but not least, the actual detected structures are placed on a solid horizontal plane corresponding to the bottom lowest altitude of the reconstruction volume. This only works well in ideal cases of cities with low varation of eleva- tion, which in addition, needs to be determined beforehand. For all these reasons, although the method can give impressive results on a dataset like Los Angeles or to a lesser extent Albuquerque, these flaws become very obvious on more chal- lenging scenes, like Four Hills, or Columbia MO. It was therefore necessary to address these weaknesses and to design an approach that could lead to denser and continuous reconstruction without making strong assumptions that are too rapidly challenged in real scenarii.

Chapter 5

Plane Sweep with Voting Pipeline

5.1 Introduction

The approach proposed in chapter 4 succeeds in triangulating an increased num- ber of detected feature points without having to use an explicit matching function. However, a significant amount of structure is still missing and some strong as- sumptions holding only in specific case had to be made to try to retrieve some of it. While the vote collapsing, or vertical extrusion is a reasonable in most case, the use of an arbitrary bottom plane as ground terrain is a weak solution that will turn out to be problematic more often than not. The approach presented in this chapter tries to adress these issues.

5.2 Pipeline Overview

The proposed pipeline in its simplest version can be decomposed into the following subparts:

- Depth Map from Stereo Vision

- Multi-View Depth Voting
- Point Cloud Extraction

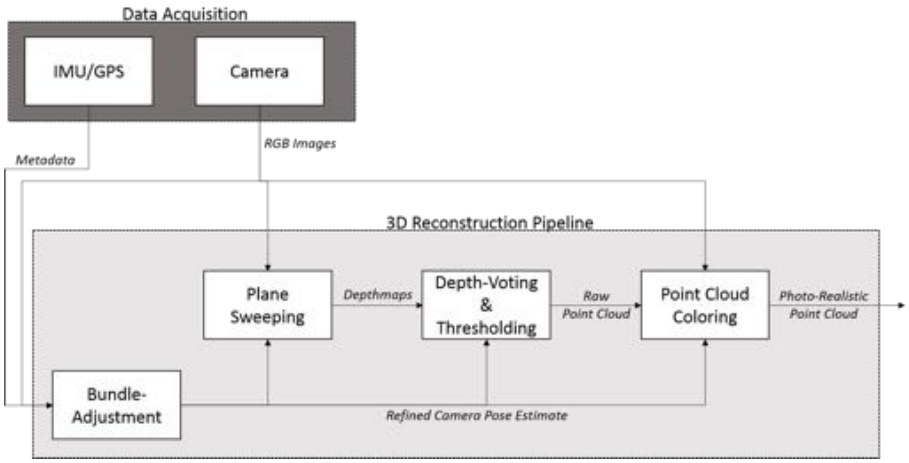


Figure 5.1: Plane Sweeping based Pipeline: This framework doesnt require the intial computation of features and instead starts with a plane sweeping phase, followed by a modified voting algorithm.

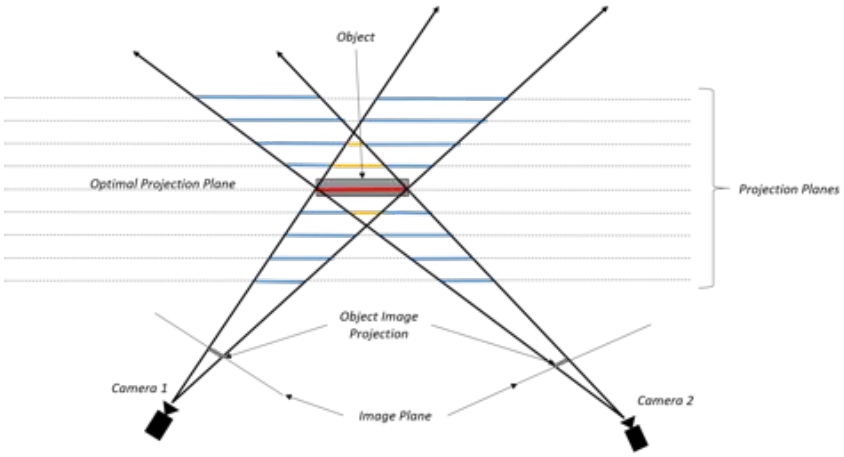


Figure 5.2: Plane sweeping principle: The projected image patch of the object, from each image, overlap perfectly only when the projection plane is at the same location as the object, in order for this method to work optimally the plane orien- tation must be parallel to the surface of the object.

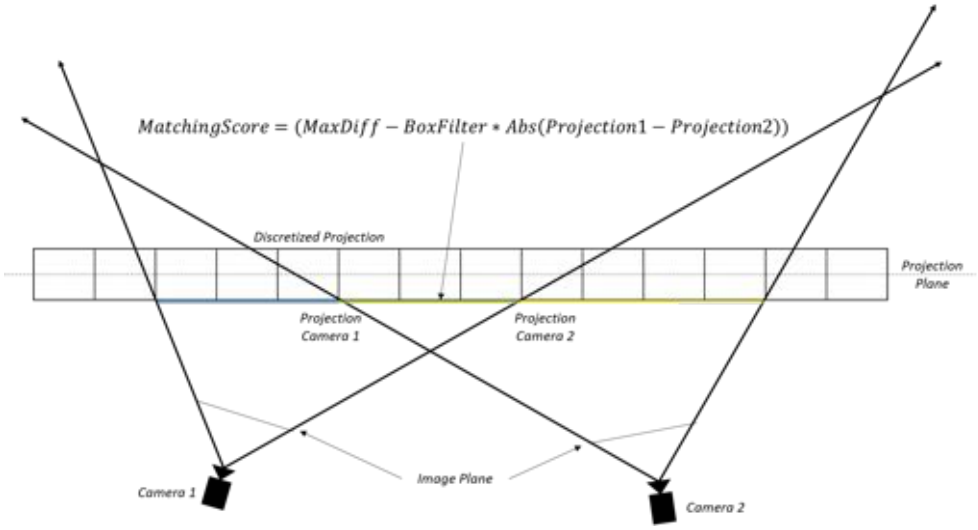


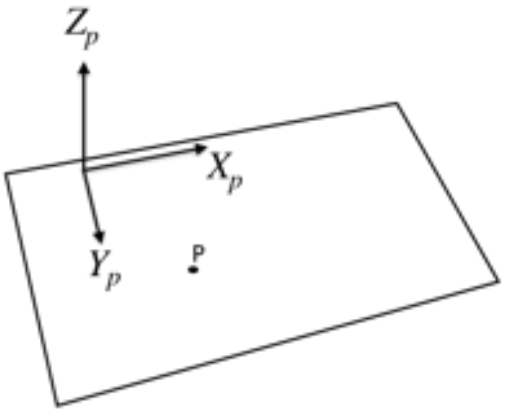
Figure 5.3: The matching only involves two views. Each image is projected on a common plane, giving two new images in a common frame of reference. A sum of absolute difference is then computed over the whole projection plane.

5.2.1 Plane Sweeping

In an attempt to achieve a complete dense reconstruction, different from the approach that consists in triangulating a limited number of points, we want to resolve continuous fields. Plane sweeping is a method that allows to compute a continuous depthmap from a pair of images by sampling the 3D space in parallel slices and finding the projection distance with maximum photometric consistency (Figure 3.2). One of the advantages of this method is that once a projection direction is defined, it is possible to compute a matching measure for several projection distance by only scaling the projected image, instead of going again through the computationaly expensive projection. Different from voting in first pipeline, the memory usage can be reduced to a minimum. Only the current projection plane and the registered maps keeping track of the maximum score and optimal distance are required to stay in memory. Parallelization is also straightforward. There are several ways to divide the work load between thread and minimize conflicts between threads for data access.

Algorithm 10 Plane sweep

$ProjectionDistance$,R_{plane}
$WarpedImage1 = BuildImageProjection(Image1, ProjectionDistance, R_{plane})$
$WarpedImage2 = BuildImageProjection(Image2, ProjectionDistance, R_{plane})$
$OptimalDistanceImage = Initialize(inf)$
$OptimalDiffImage = Initialize(inf)$
for $SweepingDistance = SweepingRangeMin : SweepingRangeMax$ **do**
  $ScaledImage1 = ScaleImage(WarpedImage1, SweepingDistance)$
  $ScaledImage2 = ScaleImage(WarpedImage2, SweepingDistance)$
  $AbsoluteDiff = Abs(ScaledImage1 - ScaledImage2)$
  $AbsoluteDiff = Abs(conv2d(ADiff, ones(KernelSize)))$
  $RescaledAbsoluteDiff = ScaleImage(AbsoluteDiff, ProjectionDistance)$
  for $pixelinOptimalDiffImage$ **do**
    if $RescaledAbsoluteDiff(pixel) < OptimalDiffImage(pixel)$ **then**
      $OptimalDiffImage(pixel) = RescaledAbsoluteDiff(pixel)$
      $OptimalDistanceImage(pixel) = ProjectionDistance$
    end if
  end for
end for



Rotation Matrix:

$$R_{plane} = \begin{bmatrix} | & | & | \\ X_p & Y_p & Z_p \\ | & | & | \end{bmatrix}_{PlaneWorld}$$

$$P_{Plane} = [x_p, y_p, z_p]^t_{Plane}$$

$$P_{World} = [x_w, y_w, z_w]_{World} = R^t_{Plane} P_{plane}$$

Figure 5.4: A local coordinate system is associated to the plane on which the image is being projected. R_{plane} allows to transform from plane coordinates to world coordinates. Then the projection matrix of a chosen camera can be used to calculate the pixel projection and obtain the color sample.

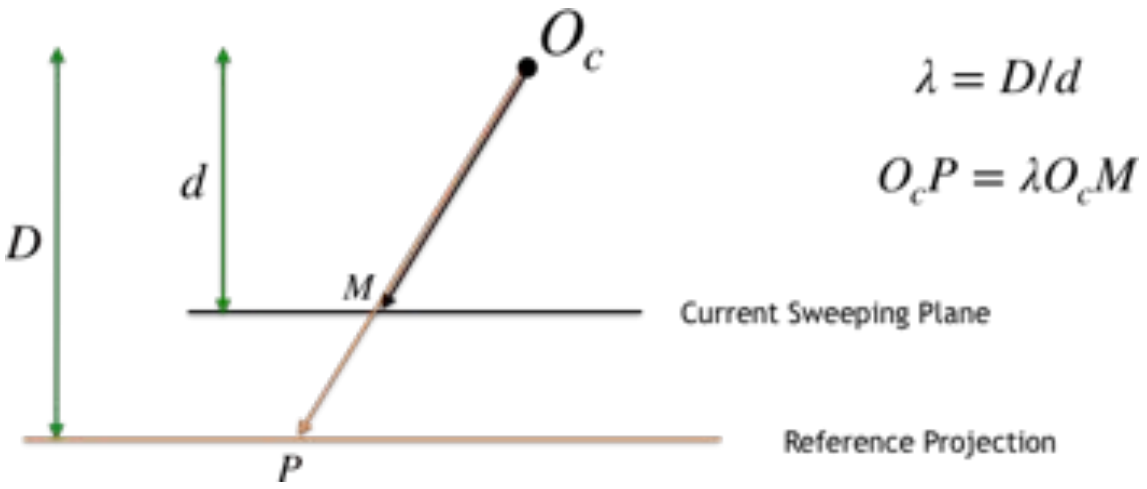


Figure 5.5: Once the image has been warped a projection on any parallel plane can be obtained by simple rescaling and translating operations.

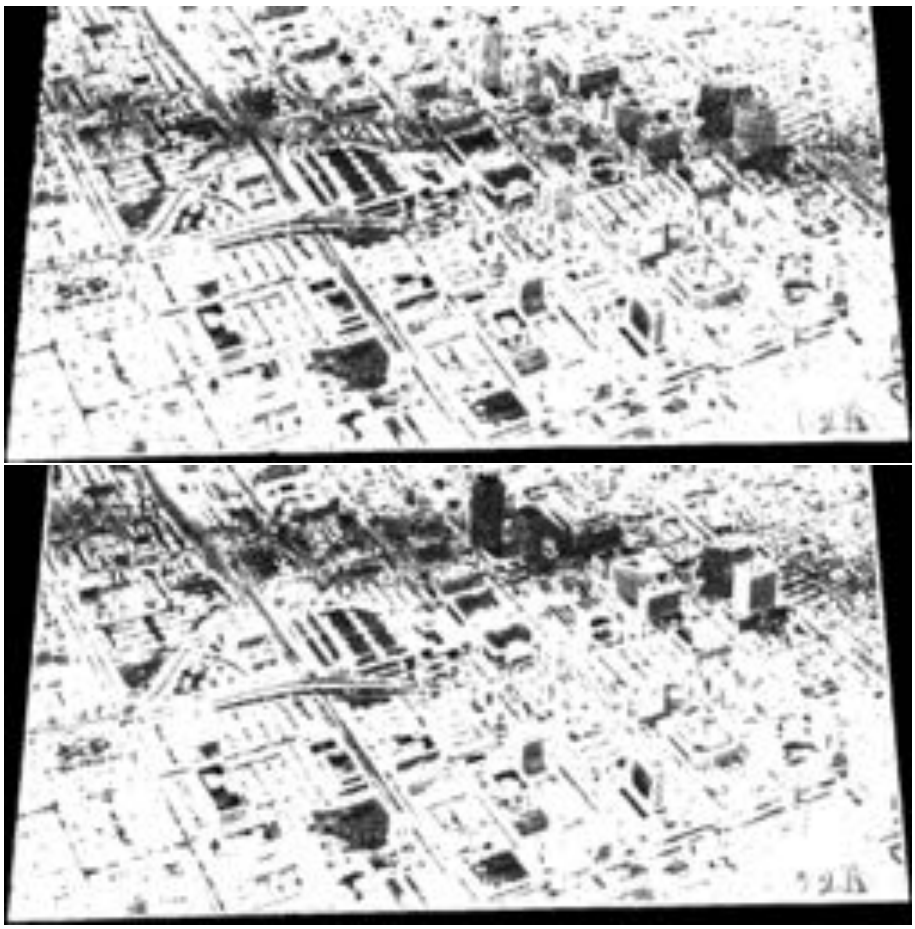
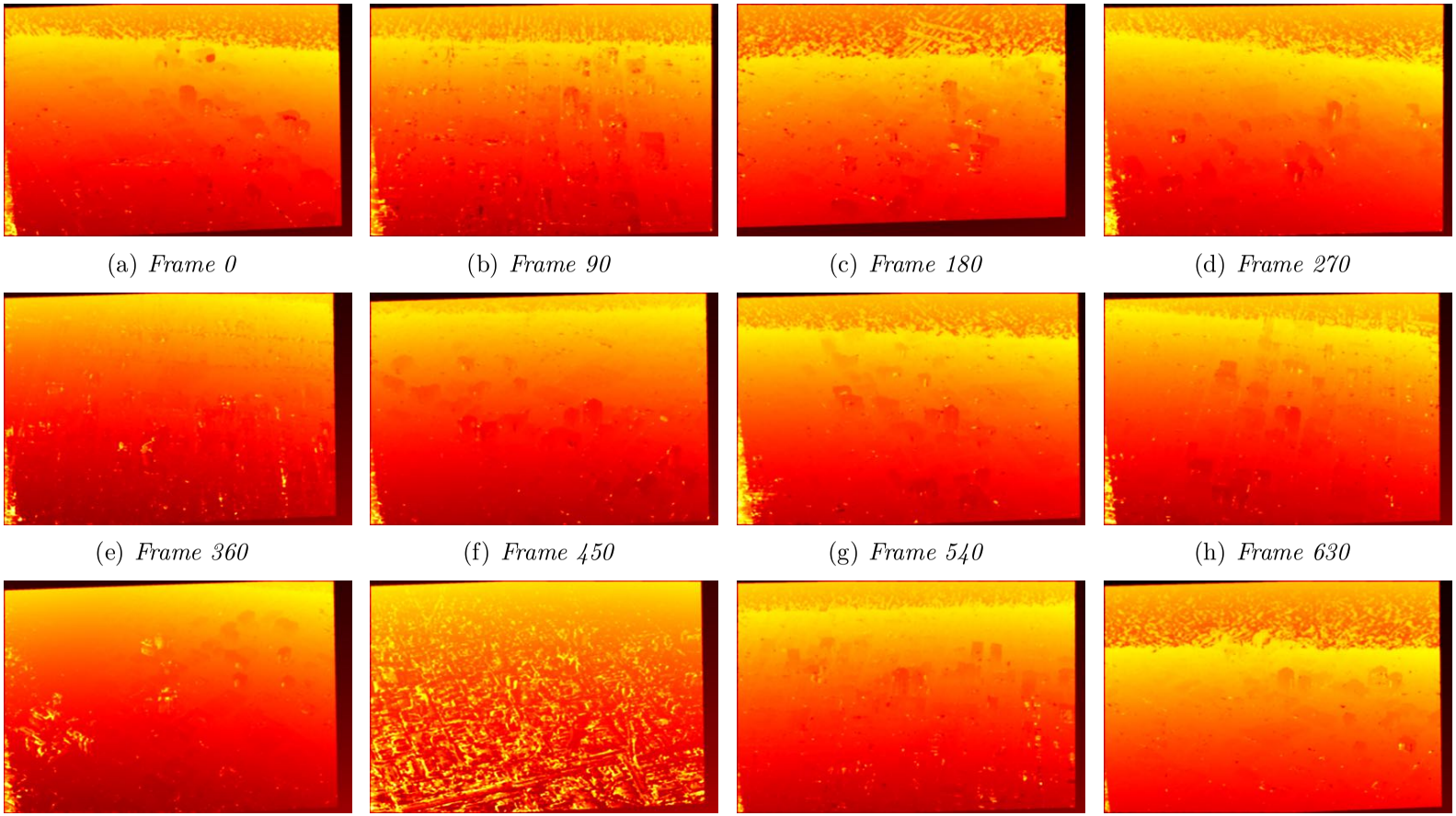


Figure 5.6: Intensity difference between two warped adjacent views at two different projection distances.



(a) *Frame 0* (b) *Frame 90* (c) *Frame 180* (d) *Frame 270*

(e) *Frame 360* (f) *Frame 450* (g) *Frame 540* (h) *Frame 630*

(i) *Frame 720* (j) *Frame 810* (k) *Frame 900* (l) *Frame 990*

Figure 5.7: Twelve output depthmaps from one orbit of Albuquerque dataset, spread over one flying orbit.

5.2.2 Depth Based Voting

Once we have produced a large enough set of Depth Map from adjacent pairs, the next step is to combined the information into a single probabilsitic volume. Each Depthmap are telling us to a certain extend what part of space is probably empty, occupied or hidden

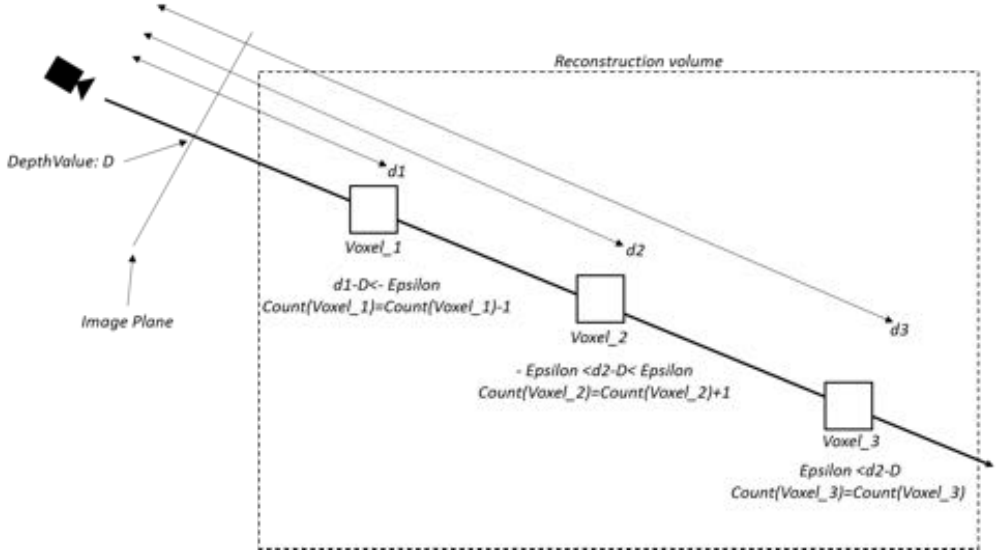


Figure 5.8: Depth based Voting: for each voxel, its distance to the camera center of projection is compared to the depth value read on the depthmap at its projection location. If the distance is significanlty shorter than the depth value, the voxel should be empty. Hence it is given a negative vote. If the distance is close enough to the depth value, the voxel is considered as occupied according to this view and its vote count is incremented positively. For distances beyond the read depth value, the voxel is not supposed to be visible and its vote count should not be updated.

5.2.3 Algorithms Memory Requirement

Ultimately, the plane sweep based pipeline is also a volumetric approach and the memory usage is therefore bound to the voxel grid resolution. A possible optimization is to discard each depthmap once it has been used to update the voxel grid, in which case only one depthmap in addition to the grid is needed at any time in memory

Algorithm 11 Depth-based voting Algorithm

```
1: for DepthMapIndex = 1 : NDepthMap do
2:    for VoxelIndex = 1 : NVoxels do
3:       (p_x, p_y) = GetVoxelProjection(DepthMapIndex, VoxelIndex)
4:       EstimatedDepth = DepthMap(DepthMapIndex)(p_x, P_y)
5:       Distance = GetDistance(VoxelIndex, DepthMapIndex)
6:       if Distance < EstimatedDepth − ε then
7:          Count(VoxelIndex) − −
8:       else if Distance < EstimatedDepth + ε then
9:          Count(VoxelIndex) + +
10:      end if
11:   end for
12: end for
```

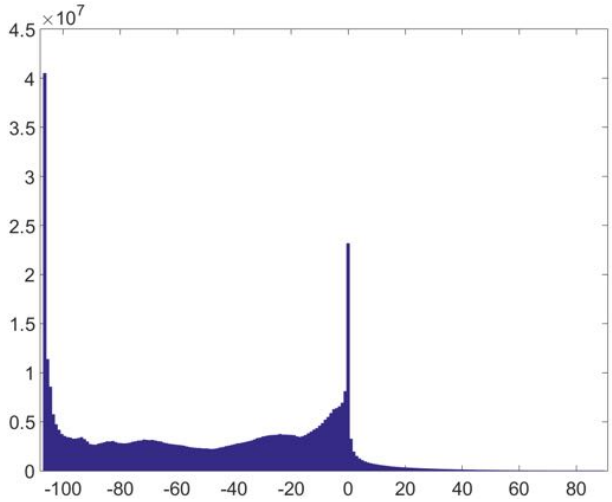
($\mathbf{O}(N_{voxels} + N_{depthmappixels})$.

5.2.4 Point Cloud Extraction

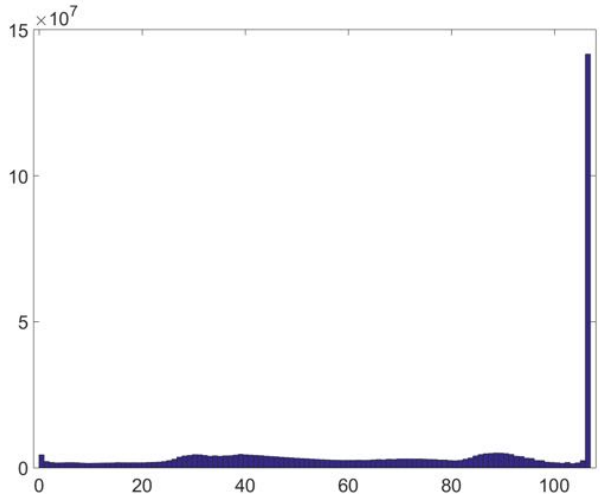
After enough depthmaps have been used for voting, a clear separation between empty space with low negative vote count and occupied space with strictly pos- itive count is expected. In that case, thresholding the vote space, unlike in our Feature Voting based pipeline, is a trivial task. Applying an extrusion step, such as the vote collapsing presented in section 2.2.2 can still help adding back missing building structures. As the method performs well overall on the terrain reconstruc- tion aspect, this therefore eliminates one of the undesirable effect of the extrusion.

5.2.5 Depth Voting and Thresholding

As a next step, the set of generated depthmap are used for voting according to the method described in the Depth based voting section. In addition to the vote count, we are keeping track for each voxel of the number of frames its projection is within the image boundaries. Figure 5.9 shows histograms of these two measures for Albuquerque reconstruction volum and Figure 5.10 shows the vote count and visibility count spaces observed from above.

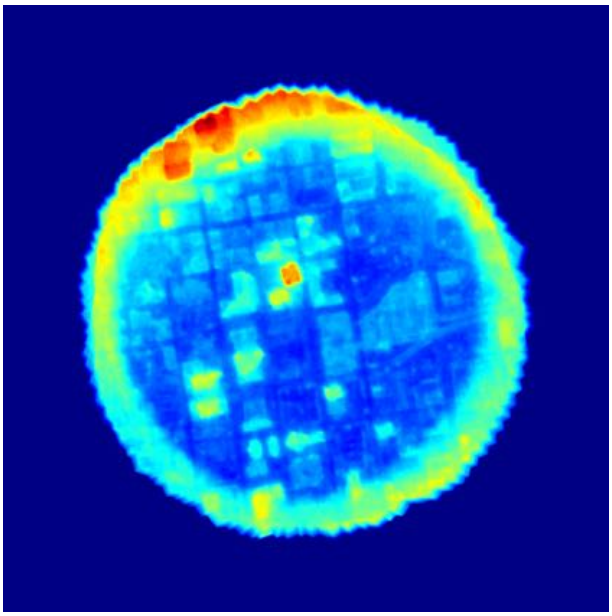


(a) *Vote Space Histogram*

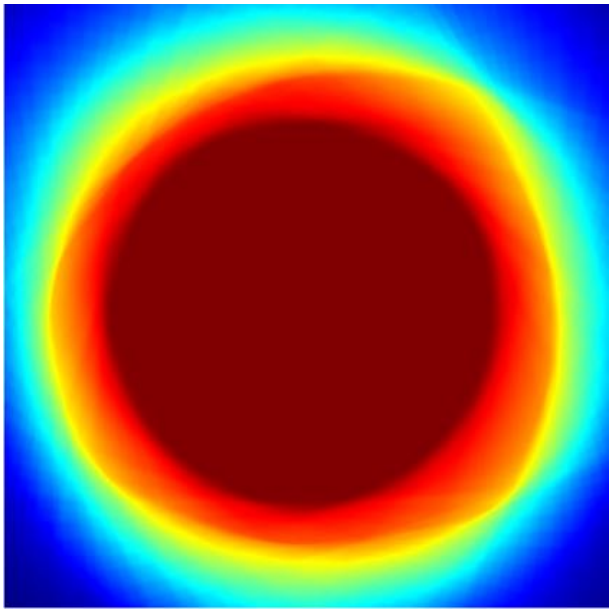


(b) *Visibility Count Histogram*

Figure 5.9: Albuquerque Vote and visibility count histograms



(a) *Vote Space visualization*

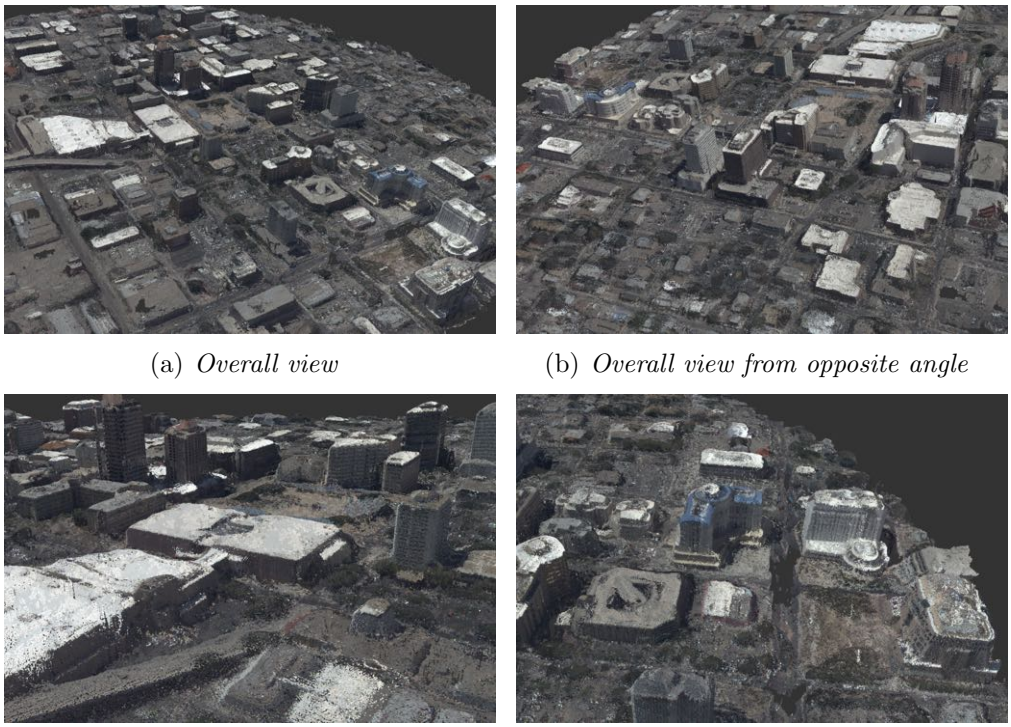


(b) *Visibility Count visualization*

Figure 5.10: Top views of vote space and visibility count for Albuquerque.

The visibility histograms and top views shape are expected. The majority of the center area is visible on every view and account for most of the space. The visibility gradually declines as we move towards the borders. Experiments also show that the empty space ends up with very low negative vote count and a trivial threshold value

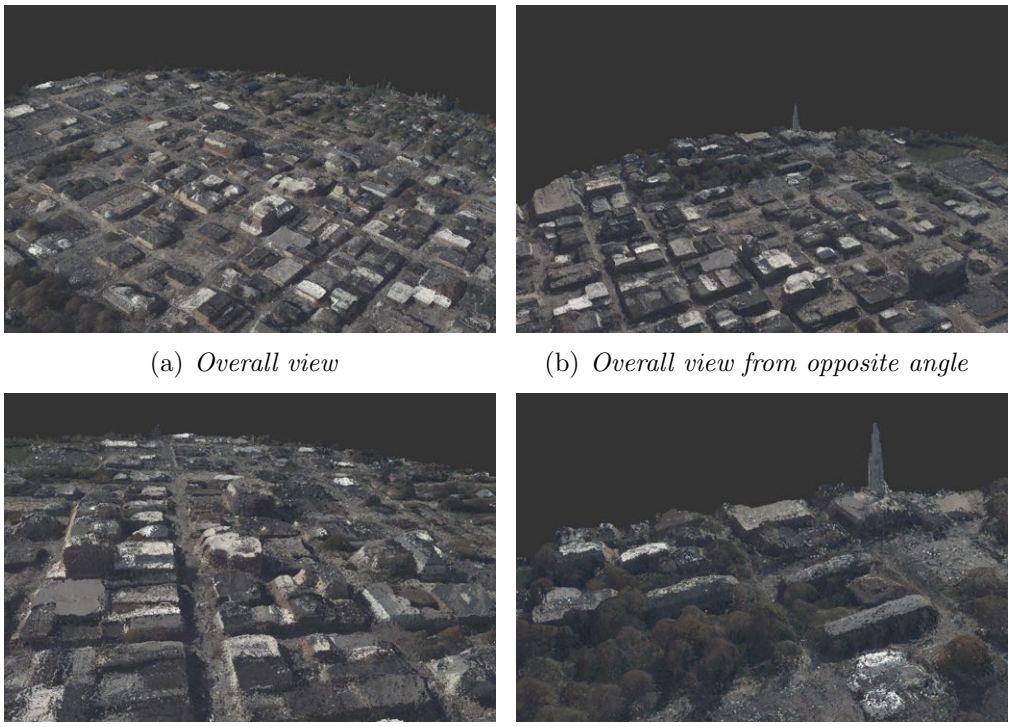
around 0 gives satisfactory result provided that we also use the visibility count to filter out voxel with uncertain vote value due lack of persistent information. After a raw point cloud has been extracted, we can reuse coloring algorithms from section **??** in order to obtain the final rendering. Figure 5.11 and 5.12 show screenshot of the final result for respectively Albuqerque and Columbia.



(a) *Overall view* (b) *Overall view from opposite angle*

(c) *Zoom on Convention Center* (d) *Zoom on Judicial Courts area*

Figure 5.11: Screenshot of final point cloud for Albuquerque



(a) *Overall view*

(b) *Overall view from opposite angle*

(c) *Zoom on Broadway street*

(d) *Zoom on 6th street, and a part of Naka Hall*

Figure 5.12: Screenshot of final point cloud for Columbia MO

successive iterations using priors

The 3D model obtained by fusing the depthmaps can be used to compute a prior and refine the sweeping range and adapt it locally on the image. The most direct way is to divide the image into tiles. (See Figure 5.13). This can potentially accelerate the computation of each depthmap in addition to making it more robust by adding information from more views.

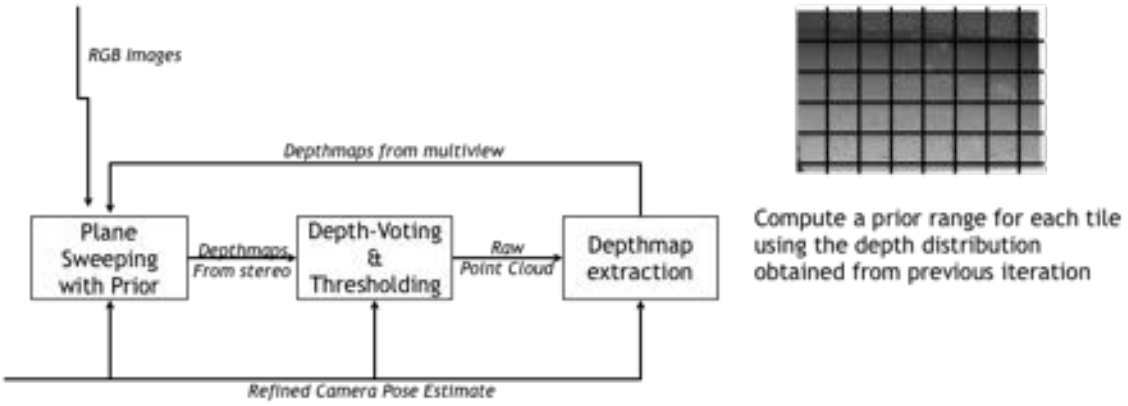


Figure 5.13: The sweeping pipeline can be initialized using a previous reconstruction result to locally refine the sweeping range. Here a regular grid is used an local prior is computed on each tile from which a more narrow sweeping range can be inferred.

Chapter 6

Benchmarking and Evaluation

This chapter discusses possible ways to quantitatively evaluate the performance of the 3D reconstruction algorithms. Benchmark datasets for 3D reconstruction exist but are not suitable for our application as Crispell et al has shown.[19] Qualitatively when looking at a reconstruction result we particularly focus on two aspects :

- The accuracy and completeness of the 3D shape
- The photoconsistency of the appearance model

Performing these evaluations qualitatively is a difficult task.

We can resort to sampling, by tracking points throughout an image sequence, and then check the variation in the position estimate. Occlusion prediction can also be evaluated that way.

Alternatively the quality of a reconstruction can be appreciated indirectly for example by assessing what was the performance gain when tracking.

6.1 Evaluation of Accuracy and Completeness of the 3D Structure

This part is the most difficult as it requires the availability of a from of Groundtruth, possibly LIDAR data or manually generated 3D models.

When this type of data are available, one can either compute a mean square error over registered depthmaps or compute difference between 3D volumes.

As a preliminary step, registration in a common coordinate system will be needed [76]. In the case of aerial imagery, it makes sense to convert the coordinates into latitude and longitude. Ideally, we would want an equation that analyticaly converts from the original reconstruction frame of coordinate to the lat-long values. If this is not possible, an alternative is to manually match at least 4 tie points from the point cloud to google map or on site measurements.

6.2 Evaluation of the Photo Consistency

This part is more straight forward. Once a photo realistic 3D model has been generated, synthetic images can be generated by specifying a camera pose identical to one of the available real images. We can then just compare the error between the two images. Photo consistency is however a measure to consider with great care. Indeed, it has already been stated that an infinity of reconstructions are photo consistent, which means that a high score in photo consistency does not necessarily mean that the 3D model is accurate. Furthermore, since the lambertian surface hypothesis does not hold in real data either, any static appearance model like the one presented in this thesis will never be able to achieve full photo-consistency as for example, reflection surfaces are not supported by this approach. Measuring photo consistency can however help setting a baseline to compare different photo realistic models as long as it is always paired with an evaluation of the shape modeling accuracy.

6.3 Qualitative Comparison

Neither 3D reconstruction nor LIDAR datasets are easily shared right now. We were able to test PMVS on transparent sky datasets. The algorithm gives reliable results sometimes better like on Columbia MO compared to the voting based pipeline, but sometimes not as good like for example on Los Angeles where the tall building facades have holes.

Deep Learning based methods do not seem for now to outperform traditional approaches. Our reconstruction results are comparable to the ones displayed by Paschalidou et al [70] (Figure 6.1).

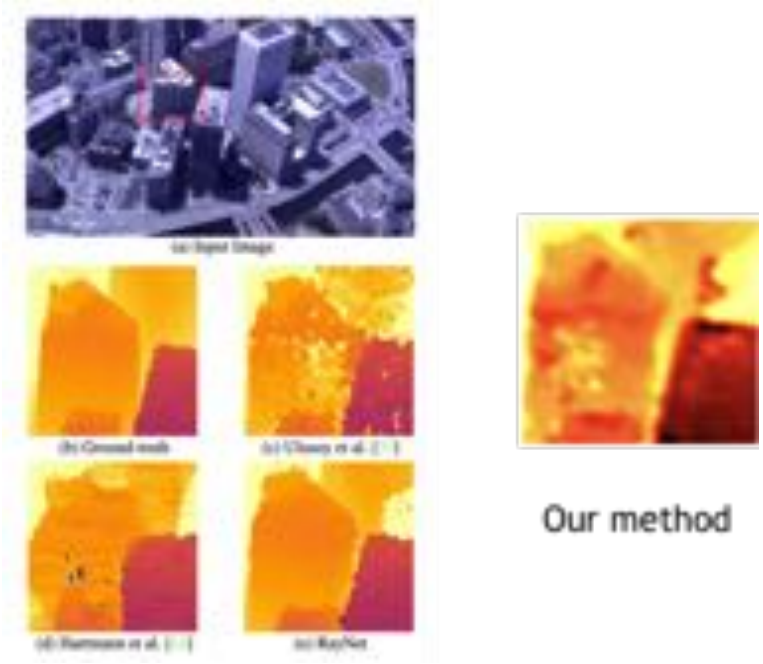


Figure 6.1: Comparison of several depthmap results from Paschalidou et al [70] and a match result of our plane sweep method.

6.3.1 Evaluation using cloud compare

More and more LIDAR data is becoming available and it was for example possible to download LIDAR data for Albuquerque and to compare it with our reconstruction

results using cloud compare [77](See Figure 6.2). The LIDAR data is however significantly more sparse than the reconstructed point cloud, which makes the interpretaton of any matching metric difficult and uncertain.

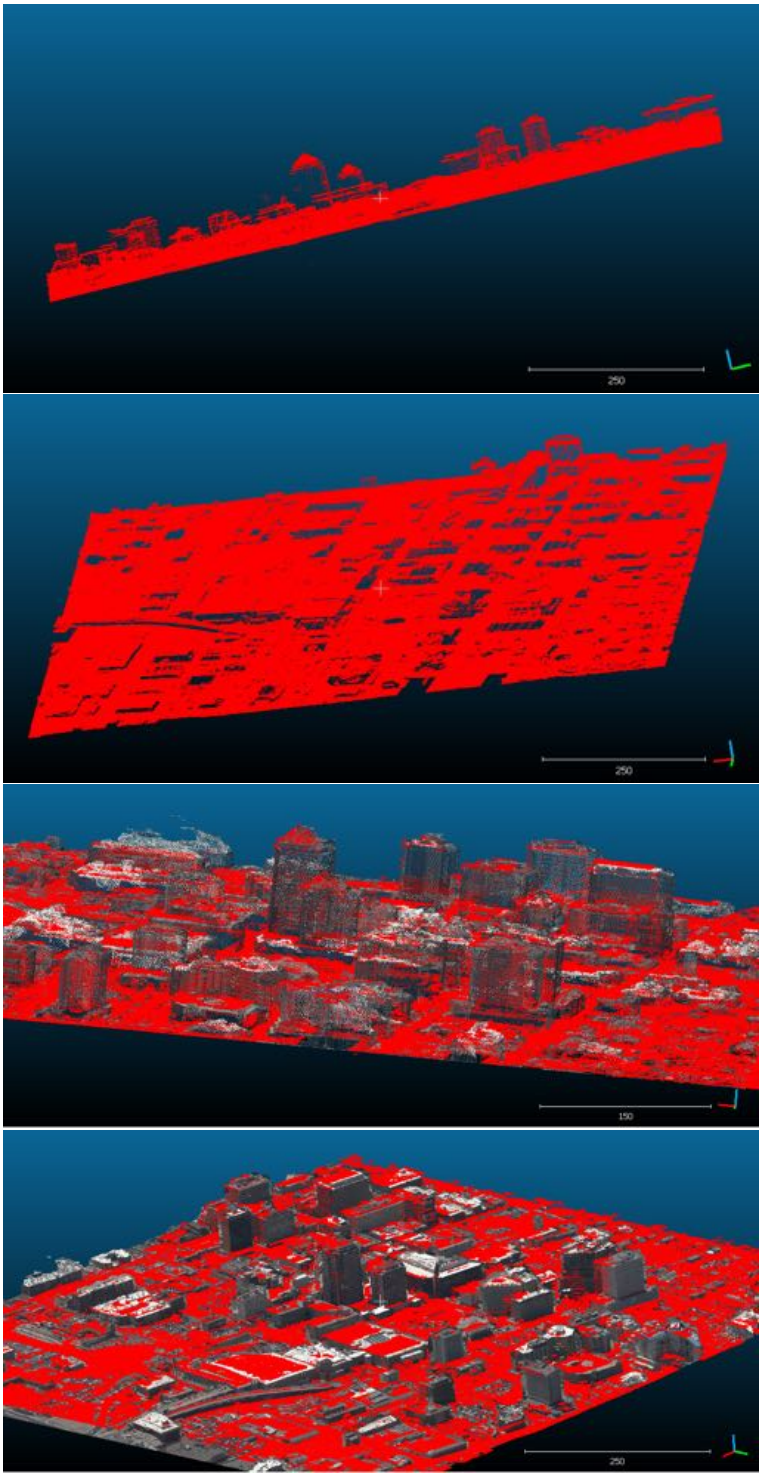


Figure 6.2: A LIDAR point cloud (in red) can be aligned with our reconstruction in order to evaluate the result in cloud compare [77].

Chapter 7

Applications

7.1 Introduction

This chapter presents a list of practical uses for the data generated by the 3D pipeline. Thus from the 3D point clouds it is possible to generate:

- Altitude Masks
- Synthetic Images
- Hazard maps
- Orthorectified maps

7.2 Altitude Masks

Object tracking in WAMI data is a very challenging task, due to lower frame rate and low definition of the object being tracked. For instance a vehicle will be fit in a window of 20 by 20 pixels and its position from one frame to another might change significantly. For these reasons it is common for any automatic tracker to

lose the target and even often continue tracking a spuriously matched image patch. Furthermore if detection relies on motion, even after registration the parallax can cause a very high number of false alarms unless additional filtering measures are taken. For that purpose, Altitude maps generated from the 3D data can prove to be very useful by filtering out impossible locations for a vehicle [78, 79].

Figures 7.2 and 7.1 show examples of altitude mask for Albuquerque.

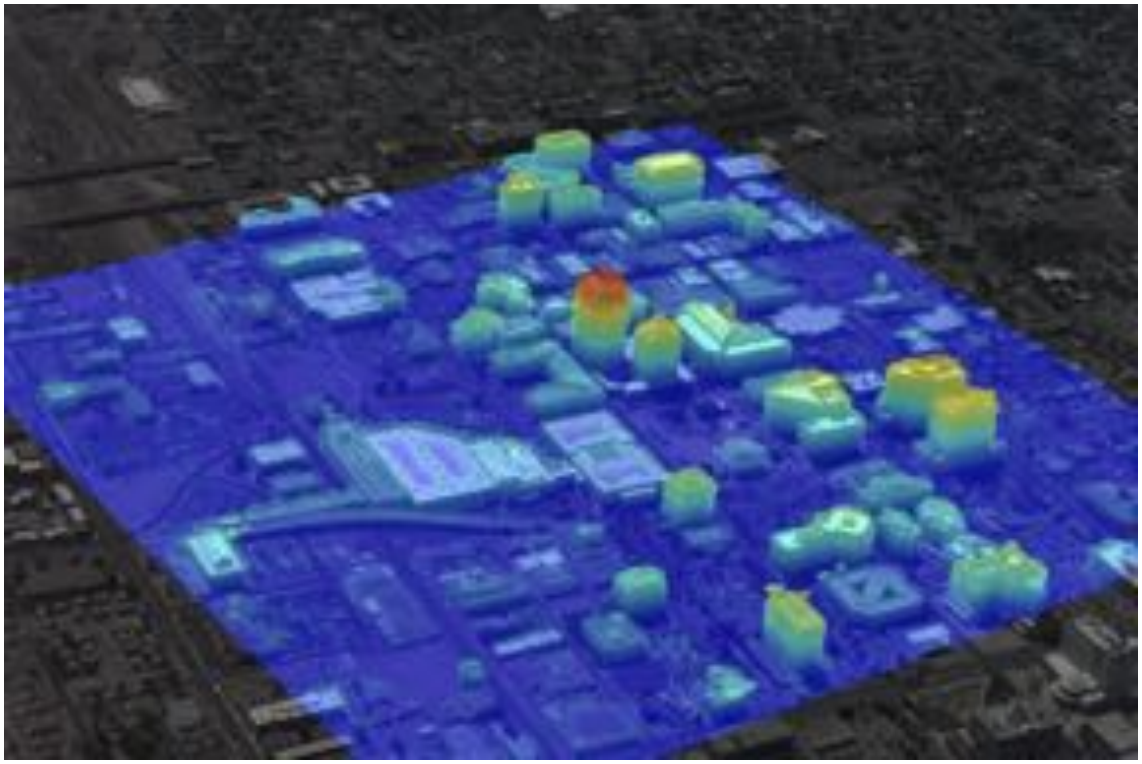


Figure 7.1: Superimposed altitude mask on its corresponding image for Albuquerque dataset.

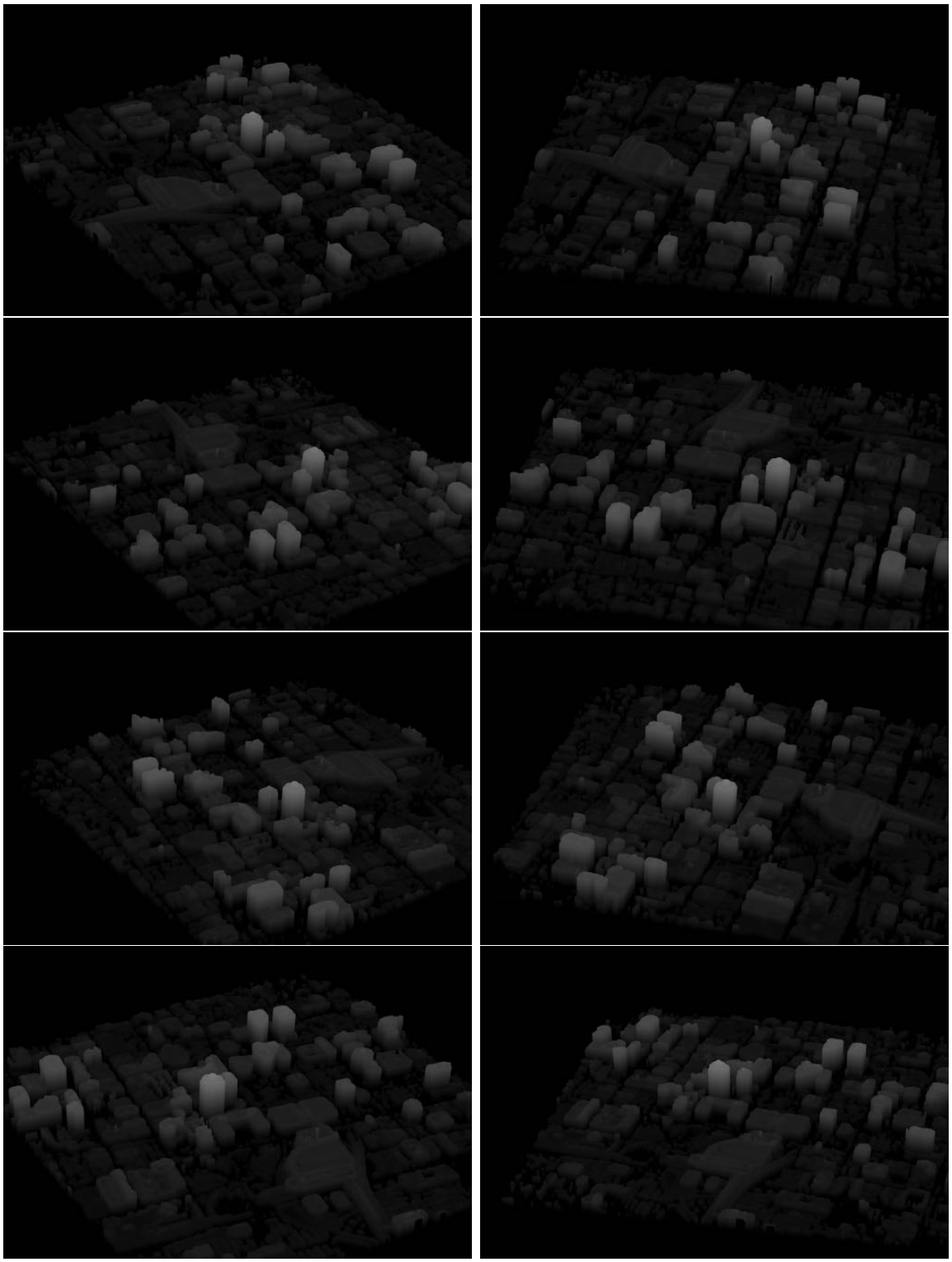


Figure 7.2: Illustration of altitude masks

7.2.1 Synthetic Images

Once we have the final output of the reconstruction, building a synthetic image is straighforward. We could project each point of the point cloud on a corresponding pixell, but if the resolution of the built image is too high, this will generate an image with most of its pixels without any color assigned. Instead we project each pixel onto an occupied voxel. (See example of synthetic image in Figure 7.3)

As an estimate of the potential compression, we can compare for Los Angeles the size of the input data to the size of the output point cloud. Los Angeles orbit sequence contains 351 frames of 6600x4400 rgb pixels which amounts if uncompressed to about 30.57 GB. The JPEG compressed images are actually taking 2.86 GB on disk. The point cloud used to generate the image from Figure 7.3 contains 4.727 millions colored points. Each points coordinates and apperance information is represented by respectively 3 floats and 3 chars, therefore taking 15 bytes in memory. The total memory space taken by the point cloud is therefore 70 MB, which represents a compression factor of around 430 with regard to the uncompressed image sequence and a factor of 40 compared to the JPEG images.

7.2.2 Hazard Map

An other application, once we have a reconsctruction is to produce an hazard or visibility map. First we need to sample the hemisphere surrounding the whole point cloud (See Figure 7.4). Each sampled point of the hemisphere is considered to be a view point. For each point in the point cloud, we compute the percentage of view points it is visible from, and eventually apply a jet colormap to the whole point cloud representing the visibility of each area (See screenshot on Figure 7.5). This visualization gives additional insight to an analyst by emphasing parts of the scenery with permanent full visibility and areas that are expected to be unaccessible on a

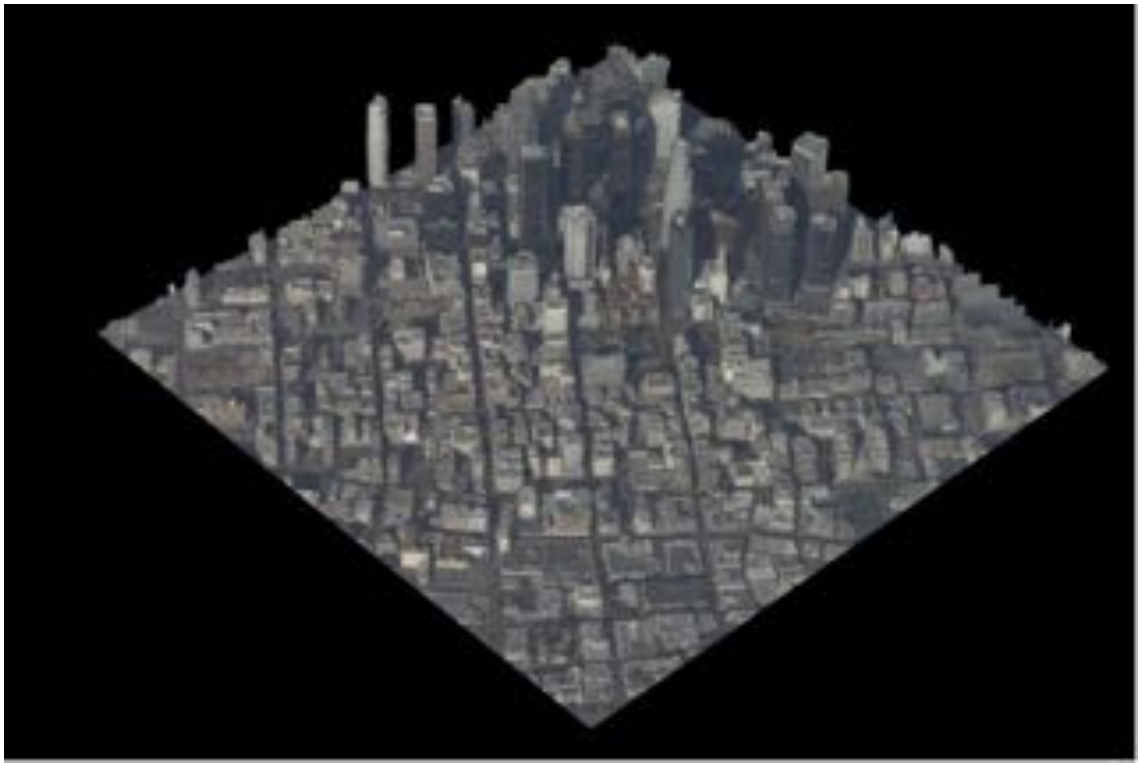


Figure 7.3: Example of synthetic images generated with Los Angeles point cloud

bigger fraction of the acquired imagery.

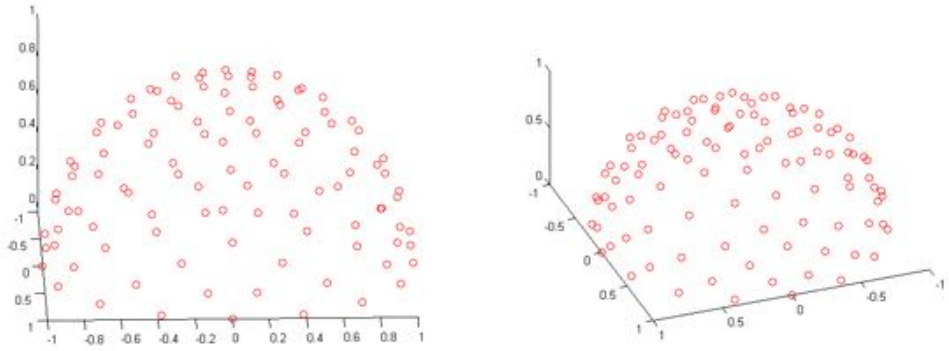


Figure 7.4: Sampling of the superior hemisphere using recursive divisions

7.3 Ortho-rectified Maps

Another possible format is an orthorectified image augmented with the 3D model that allows to identify blind spots from the current point of view. Figure 7.6 shows a

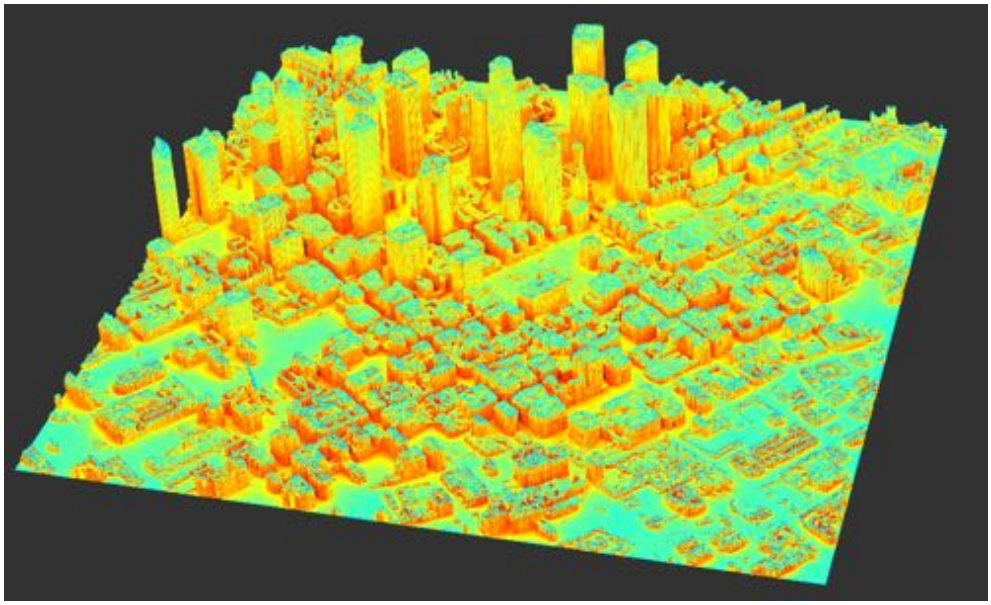


Figure 7.5: Hazard map for Los Angeles, hemisphere sampled with 105 points

preliminary result for Albuquerque of an image orthoprojection that also accounts for structural occlusions. Ortho-rectified maps can be an important part for example of a video-summarization workflow [80, 81], but the generation of a panorama without the use of 3D model will cause element of background to change from one image to another.

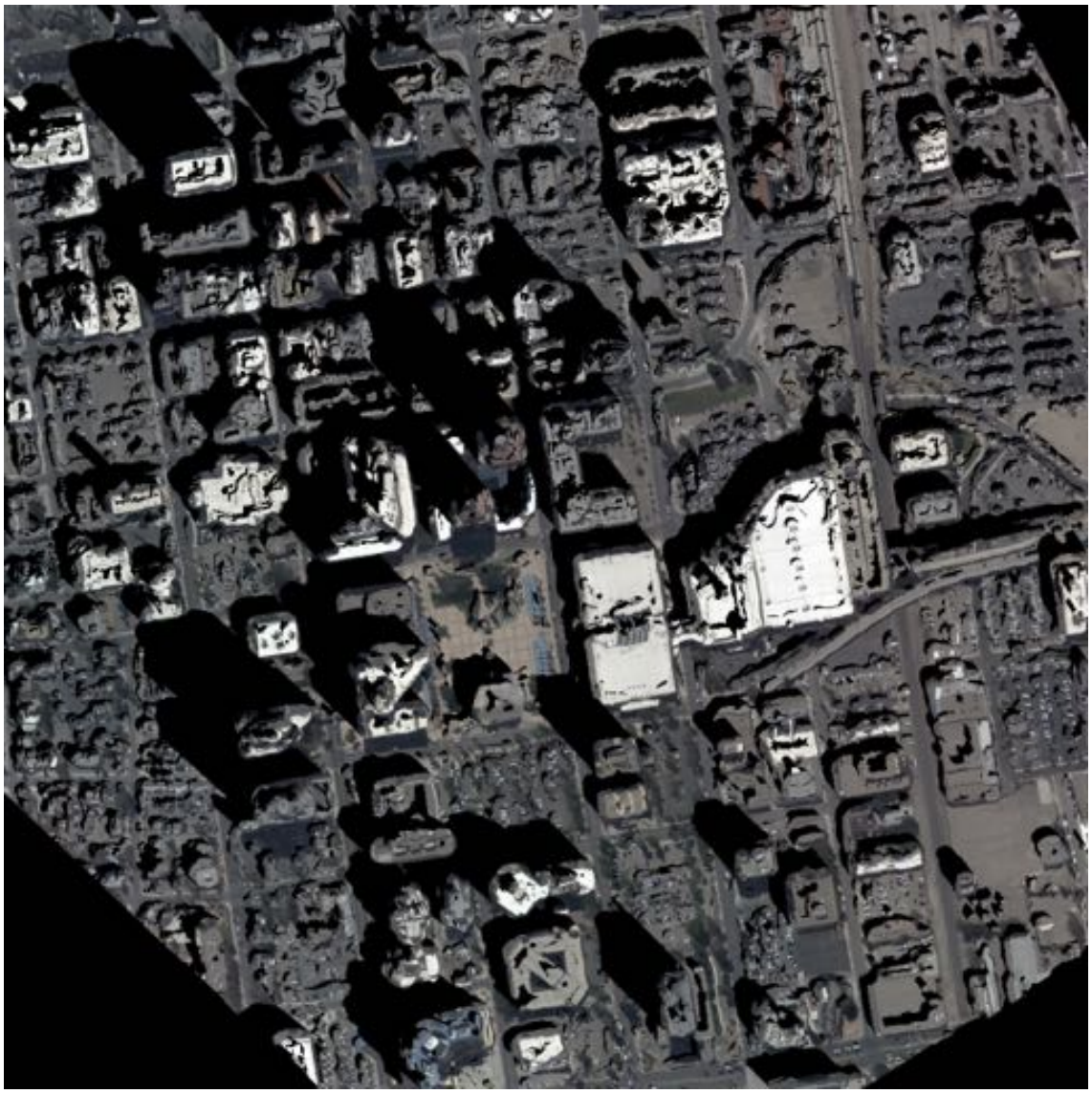


Figure 7.6: Example of 3D orthorectified image for Albuquerque. The 3D model allows to determine the occluded areas that are rendered as black shadows.

Chapter 8

Conclusion

This dissertation presents several algorithms to efficiently generate highly accurate reconstruction of city landscape from Wide Area imagery. Mainly organized in two pipelines, one based on accumulation of votes from repeatably detectable features, and one based on plane sweeping.

The algorithms have been tested on several datasets and prove to be promising allowing us to generate high quality reconstructions of very large urban areas in a very short time. Results prove still on par with Deep Neural Network based frameworks, or highly recognized methods such as PMVS.

This thesis also presents applied uses of the generated 3D data, such as:

- Hazard Maps which are an example of new information that we can learn after generating the 3D model
- Filtering through the use of altitude masks, which proved to significantly improve detection and tracking performance
- Synthetic image generation which demonstrate the potential for novel view generation and also data compression.

- Ortho rectified maps which show the potential of the data for augmented mapping.

As possible extension, the algorithms can be accelerated via parralelization, and further research could be invested into automation as it still requires educated manual interventions. The plane sweep model is extensible with adaptable local plane orientations, and different partitioning of the prior map using segmentation data.

The presented methods all, in a way or another rely on the photoconsistency assumption which in real dataset never totally holds. The inherent flaw of photo consistency was already well documented before and this research leads us to the same conclusion. The voting pipeline based on a relatively simple model, is on par with PMVS on Albuquerque dataset and gives a better result for Los Angeles. This is because the vote collapsing or vertical extrusion applies particularly well to this type of scenery. It seems even the most elegant models using only photoconsistency will expectedly meet random success or failure, while extrapolation of data via additional knowledge go a surprisingly long way towards achieving completeness. My personal belief after these years of research is that significant step up in 3D reconstruction can only be achieved by using a higher level of abstraction for example via shape recognition, either from 3D data or from images. The human brain, which is still the best computing machine at understanding its 3 dimensional surrounding does so much more than matching light intensity patterns between the two eyes. We recognize shape from our past experience or infer it from shading. Deep Neural Network having proven to be already performant on these tasks, It will probably play a decisive role in the future of 3D reconstruction too.

future work

The present work can will be further improved by:

- improve voting using available terrain information

- apply a weight to each vote depending of the length of the rays segment that actually traverses a voxel
- using 3D models from voting as initialization to the plane sweep algorithm and image segmentation to partition the image for the computation of the local priors.
- improving the plane sweep cost function to optimize by computing a multi-scale error metric, and generalize the optimization problem to more than 2 views.
- sweep along several directions and fuse the different results into a single depthmap.
- Use deep learning to try to apply scene reasoning or to design operators similar to vote collapsing in order to improve the completeness of the reconstruction.
- use input imagery from different time and fuse the separated reconstructions into a single model optimizing a confidence value over the whole set of reconstruction.

Bibliography

[1] Fabio Remondino, Luigi Barazzetti, Francesco Nex, Marco Scaioni, and Daniele Sarazzi. Uav photogrammetry for mapping and 3d modeling–current status and future perspectives. *International archives of the photogrammetry, remote sensing and spatial information sciences*, 38(1):C22, 2011.

[2] JB Antoine Maintz and Max A Viergever. A survey of medical image registration. *Medical image analysis*, 2(1):1–36, 1998.

[3] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[4] A Roy Chowdhury, Rama Chellappa, Sandeep Krishnamurthy, and Tai Vo. 3d face reconstruction from video using a generic model. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 449–452. IEEE, 2002.

[5] Erik P Blasch, Matthew Pellechia, Paul B Deignan, Kannappan Palaniappan, Shiloh L Dockstader, and Gunasekaran Seetharaman. Contemporary concerns in geographical/geospatial information systems (gis) processing. In *Aerospace and Electronics Conference (NAECON), Proceedings of the 2011 IEEE National*, pages 183–190. IEEE, 2011.

[6] Arnold Irschara, Christof Hoppe, Horst Bischof, and Stefan Kluckner. Efficient structure from motion with weak position and orientation priors. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 21–28. IEEE, 2011.

[7] Brittany Morago, Giang Bui, and Ye Duan. Integrating lidar range scans and photographs with temporal changes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 718–723, 2014.

[8] Brittany Morago. *Multi-modality fusion: registering photographs, videos, and LIDAR range scans*. University of Missouri-Columbia, 2017.

[9] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010.

[10] Kannappan Palaniappan, Filiz Bunyak, Praveen Kumar, Ilker Ersoy, Stefan Jaeger, Koyeli Ganguli, Anoop Haridas, Joshua Fraser, Raghuveer M Rao, and Guna Seetharaman. Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video. Technical report, MISSOURI UNIV-COLUMBIA DEPT OF COMPUTER SCIENCE, 2010.

[11] Rengarajan Pelapur, Kannappan Palaniappan, and Gunasekaran Seetharaman. Robust orientation and appearance adaptation for wide-area large format video object tracking. In *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*, pages 337–342. IEEE, 2012.

[12] Rengarajan Pelapur, Sema Candemir, Filiz Bunyak, Mahdieh Poostchi, Guna Seetharaman, and Kannappan Palaniappan. Persistent target tracking using likelihood fusion in wide-area and full motion video sequences. In *Information*

Fusion (FUSION), 2012 15th International Conference on, pages 2420–2427. IEEE, 2012.

[13] Kannappan Palaniappan, Raghuveer M Rao, and Guna Seetharaman. Wide-area persistent airborne video: Architecture and challenges. In *Distributed video sensor networks*, pages 349–371. Springer, 2011.

[14] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[15] Johannes L Schönberger, Friedrich Fraundorfer, and Jan-Michael Frahm. Structure-from-motion for mav image sequence analysis with photogrammetric applications. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2014.

[16] Simon Lynen, Torsten Sattler, Michael Bosse, Joel A Hesch, Marc Pollefeys, and Roland Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems*, 2015.

[17] Rahul Bhotika, David J Fleet, and Kiriakos N Kutulakos. A probabilistic theory of occupancy and emptiness. In *European conference on computer vision*, pages 112–130. Springer, 2002.

[18] Rahul Bhotika. *Scene-space methods for bayesian inference of 3d shape and motion*. PhD thesis, PhD thesis, University of Rochester, New York, 2003. Supervisor-Kiriakos N. Kutulakos. 61 155 156 BIBLIOGRAPHY, 2004.

[19] Daniel E Crispell. *A continuous probabilistic scene model for aerial imagery*. PhD thesis, Brown University, 2010.

[20] Illusionpoint.com.

[21] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[22] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012.

[23] Reimar Lenz. Linsenfehlerkorrigierte eichung von halbleiterkameras mit standardobjektiven für hochgenaue 3dmessungen in echtzeit. In *Mustererkennung 1987*, pages 212–216. Springer, 1987.

[24] Jason P De Villiers, F Wilhelm Leuschner, and Ronelle Geldenhuys. Centi-pixel accurate real-time inverse distortion correction. In *Optomechatronic Technologies 2008*, volume 7266, page 726611. International Society for Optics and Photonics, 2008.

[25] Timothy A Clarke and John G Fryer. The development of camera calibration methods and models. *The Photogrammetric Record*, 16(91):51–66, 1998.

[26] Elsayed E Hemayed. A survey of camera self-calibration. In *Advanced Video and Signal Based Surveillance, 2003. Proceedings. IEEE Conference on*, pages 351–357. IEEE, 2003.

[27] Matthew Brown, David G Lowe, et al. Recognising panoramas. In *ICCV*, volume 3, page 1218, 2003.

[28] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[29] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[30] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[31] http://graphics.stanford.edu/data/3dscanrep/.

[32] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[33] Niko Sünderhauf and Peter Protzel. Towards using sparse bundle adjustment for robust stereo odometry in outdoor terrain. 2006.

[34] Aleksandr Y Aravkin, Michael Styer, Zachary Moratto, Ara Nefian, and Michael Broxton. Student's t robust bundle adjustment algorithm. *arXiv preprint arXiv:1111.1400*, 2011.

[35] Andrea Albarelli, Emanuele Rodolà, and Andrea Torsello. Imposing semi-local geometric constraints for accurate correspondences selection in structure from motion: A game-theoretic perspective. *International journal of computer vision*, 97(1):36–53, 2012.

[36] H. AliAkbarpour, K. Palaniappan, and G. Seetharaman. Fast structure from motion for sequential and wide area motion imagery. Dec 2015.

[37] Hadi AliAkbarpour, Kannappan Palaniappan, and Guna Seetharaman. Parallax-tolerant aerial image georegistration and efficient camera pose refinementwithout

piecewise homographies. *IEEE Transactions on Geoscience and Remote Sensing*, 55(8):4618–4637, 2017.

[38] Hadi Aliakbarpour, Kannappan Palaniappan, and Guna Seetharaman. Robust camera pose refinement and rapid sfm for multiview aerial imagerywithout ransac. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2203–2207, 2015.

[39] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011.

[40] Karthikeyan Natesan Ramamurthy, Chung-Ching Lin, Aleksandr Aravkin, Sharath Pankanti, and Raphael Viguier. Distributed bundle adjustment. *arXiv preprint arXiv:1708.07954*, 2017.

[41] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[42] Jens Ackermann, Michael Goesele, et al. A survey of photometric stereo techniques. *Foundations and Trends® in Computer Graphics and Vision*, 9(3-4):149–254, 2015.

[43] Sudipta N Sinha, Daniel Scharstein, and Richard Szeliski. Efficient high-resolution stereo matching using local plane sweeps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1582–1589, 2014.

[44] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[45] Heiko Hirschmuller. Stereo vision in structured environments by consistent semi-global matching. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2386–2393. IEEE, 2006.

[46] Heiko Hirschmüller. Semi-global matching-motivation, developments and applications. In *Photogrammetric week*, volume 11, pages 173–184, 2011.

[47] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519–528. IEEE, 2006.

[48] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[49] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2010.

[50] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):903–920, 2012.

[51] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (ToG)*, 28(3):24, 2009.

[52] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International journal of computer vision*, 38(3):199–218, 2000.

[53] Annie Yao and Andrew D Calway. Dense 3-d structure from image sequences using probabilistic depth carving. In *BMVC*, volume 1, page 2, 2003.

[54] Adrian Broadhurst, Tom W Drummond, and Roberto Cipolla. A probabilistic framework for space carving. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 388–393. IEEE, 2001.

[55] Greg Slabaugh, Ron Schafer, Tom Malzbender, and Bruce Culbertson. A survey of methods for volumetric scene reconstruction from photographs. In *Volume Graphics 2001*, pages 81–100. Springer, 2001.

[56] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on pattern analysis and machine intelligence*, 16(2):150–162, 1994.

[57] Hadi Aliakbarpour and Jorge Dias. Phd forum: Volumetric 3d reconstruction without planar ground assumption. In *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, pages 1–2. IEEE, 2011.

[58] Thomas Pollard and Joseph L Mundy. Change detection in a 3-d world. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. Ieee, 2007.

[59] Thomas B Pollard. *Comprehensive three dimensional change detection using volumetric appearance modeling*. PhD thesis, Brown University, 2009.

[60] Ali Osman Ulusoy, Andreas Geiger, and Michael J Black. Towards probabilistic volumetric reconstruction using ray potentials. In *3D Vision (3DV), 2015 International Conference on*, pages 10–18. IEEE, 2015.

[61] Fatih Calakli and Gabriel Taubin. Ssd: Smooth signed distance surface reconstruction. In *Computer Graphics Forum*, volume 30, pages 1993–2002. Wiley Online Library, 2011.

[62] Fatih Calakli, Ali O Ulusoy, Maria I Restrepo, Gabriel Taubin, and Joseph L Mundy. High resolution surface reconstruction from multi-view aerial imagery. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 25–32. IEEE, 2012.

[63] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999.

[64] Erwan Guillou, Daniel Meneveaux, Eric Maisel, and Kadi Bouatouch. Using vanishing points for camera calibration and coarse 3d reconstruction from a single image. *The Visual Computer*, 16(7):396–410, 2000.

[65] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018.

[66] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[67] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 38, page 1, 2017.

[68] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016.

[69] J Gwak, Christopher B Choy, Animesh Garg, Manmohan Chandraker, and Silvio Savarese. Weakly supervised generative adversarial networks for 3d reconstruction. *CoRR, vol. abs/1705.10904*, 2, 2017.

[70] Despoina Paschalidou, Ali Osman Ulusoy, Carolin Schmitt, Luc Van Gool, and Andreas Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3897–3906, 2018.

[71] https://www.transparentsky.net/.

[72] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *European conference on computer vision*, pages 128–142. Springer, 2002.

[73] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[74] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[75] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[76] Maria I Restrepo, Ali O Ulusoy, and Joseph L Mundy. Evaluation of feature-based 3-d registration of probabilistic volumetric scenes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 98:1–18, 2014.

[77] https://www.danielgm.net/cc/.

[78] Mahdieh Poostchi, Hadi Aliakbarpour, Raphael Viguier, Filiz Bunyak, Kannappan Palaniappan, and Guna Seetharaman. Semantic depth map fusion for

moving vehicle detection in aerial video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 32–40, 2016.

[79] Kannappan Palaniappan, Mahdieh Poostchi, Hadi Aliakbarpour, Raphael Viguier, Joshua Fraser, Filiz Bunyak, Arslan Basharat, Steve Suddarth, Erik Blasch, Raghuveer M Rao, et al. Moving object detection for vehicle tracking in wide area motion imagery using 4d filtering. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2830–2835. IEEE, 2016.

[80] Raphael Viguier, C-C Lin, Karthik Swaminathan, Augusto Vega, Alper Buyuktosunoglu, Sharathchandra Pankanti, Pradip Bose, H Akbarpour, Filiz Bunyak, Kannappan Palaniappan, et al. Resilient mobile cognition: Algorithms, innovations, and architectures. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 728–731. IEEE, 2015.

[81] Raphael Viguier, Chung Ching Lin, Hadi AliAkbarpour, Filiz Bunyak, Sharathchandra Pankanti, Guna Seetharaman, and Kannappan Palaniappan. Automatic video content summarization using geospatial mosaics of aerial imagery. In *Multimedia (ISM), 2015 IEEE International Symposium on*, pages 249–253. IEEE, 2015.

VITA

Raphael Viguier was born in 1987 in Versailles, France. After obtaining his Scientifique Baccalaurat in Villaroy, Guyancourt in 2004, he attended Lycee Michelet's Mathematique Superieure and Speciale classes, in Vanves, before being admitted at Ecole Centrale de Lille(ECL) in summer 2007. He graduated from ECL in 2011 with a Masters Degree of Engineering. Raphael first worked under Dr. Palaniappan supervision as a research intern during summer 2011 as a part of his Masters degree requirement. He later started a doctorate degree at University of Missouri in January 2012. His research focused on 3D reconstruction in Wide Area Imagery, Tracking and Data fusion. During his doctorate degree, Raphael worked as a research intern at IBM Watson Researching Yorktown Heights NY from March 2015 to December 2016. At IBM, he worked on Video-Summarization in Wide Area Motion Imagery, and Tracking on full motion videos from law enforcement body-worn cameras. In January 2018 he started a new position as Computer Vision Engineer at Numina, in NYC. He received his Ph.D. degree in computer science from the University of Missouri Columbia in July 2018