SUPER-RESOLUTION TEXTURED DIGITAL SURFACE MAP (DSM) FORMATION BY SELECTING THE TEXTURE FROM MULTIPLE PERSPECTIVE TEXEL IMAGES TAKEN BY A LOW-COST SMALL UNMANNED AERIAL VEHICLE (UAV)

by

Bikalpa Khatiwada

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

Scott E. Budge, Ph.D. Major Professor	Jacob H. Gunther, Ph.D. Committee Member
Rose Hu, Ph.D. Committee Member	Todd K. Moon, Ph.D. Committee Member
Xiaojun Qi, Ph.D. Committee Member	Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2023

ABSTRACT

Super-resolution Textured Digital Surface Map (DSM) formation
by selecting the texture from multiple perspective texel
images taken by a low-cost small Unmanned
Aerial Vehicle (UAV)

by

Bikalpa Khatiwada, Doctor of Philosophy
Utah State University, 2023

Major Professor: Scott E. Budge, Ph.D.
Department: Electrical and Computer Engineering

Textured Digital Surface Model (TDSM) has been used in many applications due to its ability to offer significant insights into the topographical features of a terrain and enable better visualization in various applications. A texel camera is a sensor package comprising a lidar camera, a digital camera and an inertial sensor. Texel images captured by a texel camera are used to create a TDSM. TDSM formation using a texel camera mounted on a small unmanned aerial vehicle is significantly cheaper to implement compared to the manned aircraft-based methods.

This dissertation presents the two main improvements over the existing TDSM generation using texel images. Firstly, it describes the algorithm needed to implement the optimization in a streaming fashion so that it can handle a large number of texel images. The registration of several input texel images is done by reading a window of texel images, optimizing the registration parameters, and adding them to the TDSM structure. This makes it possible to create a TDSM using many input texel images that cover a large area.

The other improvement includes making the texture of the TDSM better. This is done by selecting optimal texture fragments to texture the surface model. Various criteria are implemented to select the optimal texture. A novel packing algorithm is also presented to texture these fragments on a single output image canvas. Finally, a super-resolution algorithm is also implemented to increase the resolution of the triangle fragments. A spatially varying Wiener filter which is a minimum mean square filter is implemented to deconvolve many low-resolution triangles into a high-resolution triangle. Analysis of the results using various datasets with these algorithms is presented. The final output TDSMs are shown and the improvement of the texture over the previous method is reported.

(124 pages)

PUBLIC ABSTRACT

Super-resolution Textured Digital Surface Map (DSM) formation
by selecting the texture from multiple perspective texel
images taken by a low-cost small Unmanned
Aerial Vehicle (UAV)

Bikalpa Khatiwada

Textured Digital Surface Model (TDSM) is a three-dimensional terrain map with texture overlaid on it. Utah State University has developed a texel camera which can capture a 3D image called a texel image. A TDSM can be constructed by combining these multiple texel images, which is much cheaper than the traditional method. The overall goal is to create a TDSM for a larger area that is cheaper and equally accurate as the TDSM created using a high-cost system.

The images obtained from such an inexpensive camera have a lot of errors. To create scientifically accurate TDSM, the error presented in the image must be corrected. An automatic process to create TDSM is presented that can handle a large number of input texel images. The advantage of using such a large set of input images is that they can cover a large area on the ground, making the algorithm suitable for large-scale applications. This is done by processing images and correcting them in a windowing manner. Furthermore, the appearance of the final 3D terrain map is improved by selecting the texture from many candidate images. This ensures that the best texture is selected. The selection criteria are discussed. Lastly, a method to increase the resolution of the final image is discussed. The methods described in this dissertation improve the current technique of creating TDSM, and the results are shown and analyzed.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude toward my major professor, Dr. Scott Budge, for his invaluable guidance, support, and encouragement throughout this research journey. His insights and suggestions have helped me shape my research. I would also like to thank my committee members, Dr. Jacob Gunther, Dr.Todd Moon, Dr. Rose Hu, and Dr. Xiaojun Qi, for their support and feedback that helped me accomplish this research study. I am thankful to all my fellow students in the Center for Advanced Imaging Ladar over the years, especially Xuan Xie and Trevor Welch.

Most of all, I cannot begin to thank my wife, Sierra Weeks, for all her sacrifice, support and love throughout this academic journey. Her patience and understanding have been pivotal in completing my graduate degree. I would also like to acknowledge my parents, Yub Raj Khatiwada and Mira Neupane, and my brother, Bipin Khatiwada, for their unwavering support and encouragement throughout my life. I like to thank all my Nepalese friends in Logan whose love and motivation have been a constant source of inspiration. Lastly, I would like a special thanks to Wasim Akram Khan for his valuable advice on my work.

Bikalpa Khatiwada

CONTENTS

Page

ABSTRACT . . . iii

PUBLIC ABSTRACT . . . v

ACKNOWLEDGMENTS . . . vi

LIST OF TABLES . . . ix

LIST OF FIGURES . . . x

ACRONYMS . . . xiii

1 Introduction . . . 1
- 1.1 Statement of the Problem . . . 4
- 1.2 Research aims, objectives, and research question . . . 6
- 1.3 Significance of the study and limitations . . . 7
- 1.4 Outline . . . 7

2 Texel Camera and Fundamental definition . . . 9
- 2.1 Coboresighted Texel Camera . . . 9
- 2.2 Bistatic Texel Camera . . . 10
- 2.3 Texel Image . . . 10
- 2.4 Quaternion . . . 12
- 2.5 Camera pose and transformation between coordinate frames . . . 15
- 2.6 Normalized plane . . . 17
- 2.7 Projection and range coordinate system . . . 18
- 2.8 Calibration of bistatic sensor and parallax correction . . . 19
- 2.9 Formation of the Projection matrix . . . 22

3 Streaming Bundle Adjustment . . . 24
- 3.1 Introduction . . . 24
- 3.2 Texel Registration . . . 27
- 3.3 Conventional Bundle Adjustment . . . 31
- 3.4 Streaming Bundle Adjustment . . . 34
- 3.5 Illustration example . . . 36
- 3.6 Results and Discussion . . . 41

4 Selection and Packing of Optimal Texture Fragments . . . 51
- 4.1 Introduction . . . 51
- 4.2 Registration of Texel Images . . . 53
- 4.3 Texture Selection . . . 54
- 4.4 Texture Packing . . . 57

4.4.1 Grouping triangles into rectangles . . . 57
4.4.2 Packing rectangle into a canvas(output texture) . . . 59
4.5 Result and Discussion . . . 61

5 Superresolution . . . 74
5.1 Adaptive Wiener Filter . . . 78
5.2 Methodology . . . 78
5.3 Spatially-Varying Statistical model . . . 83
5.4 Results and Discussion . . . 88

6 Conclusion and Future Work . . . 101
6.1 Conclusion . . . 101
6.2 Future Work . . . 102

REFERENCES . . . 104

CURRICULUM VITAE . . . 111

LIST OF TABLES

Table Page

2.1 An example of a projection matrix 23

3.1 Projection Matrix for CBA 37

3.2 Jacobian matrix for CBA 38

3.3 Projection matrix for SBA, Iteration 1 39

3.4 Projection matrix for SBA, Iteration 2 40

3.5 Projection matrix for SBA, Iteration 3 41

3.6 Jacobian matrix for SBA, Iteration 2 41

3.7 Jacobian matrix for SBA, Iteration 3 42

3.8 Error comparison between non-registered TDSM and TDSM created using SBA with different $\mathcal{L}$ 45

3.9 Memory consumption comparison between SBA and CBA for different *look length* 46

4.1 Table showing number of pixel in existing method and the 66

4.2 Comparison of the final image size between initial, base packing, and splitting canvas with iterative algorithm is applied 66

LIST OF FIGURES

Figure Page

1.1 An example flight scenario: a model of texel camera taking images of the terrain. . . . 3

1.2 Simulated TDSM generated from the example flight above 3

1.3 3D triangle (marked with yellow) with six different candidate images 5

2.1 Ray diagram of a coboresighted texel camera. . . . 10

2.2 Second generation coboresighted texel camera. . . . 11

2.3 Bistatic texel camera model. . . . 11

2.4 Bistatic texel camera developed by CAIL. . . . 12

2.5 Rotation representation. . . . 13

2.6 Aligninment of both sensors such that their principal rays are parallel. . . . 20

3.1 Flowchart showing registration of texel images using SBA algorithm 26

3.2 SBA operation 35

3.3 (a) Texel camera (b) Texel camera mounted in DJI rotorcraft for data acquisition 42

3.4 Registration results (a) Before registration (b) Registration using CBA (c) Registration using SBA 47

3.5 Registration results (a) Before registration (b) Registration using CBA (c) Registration using SBA 48

3.6 TDSM with different *look length* 49

3.7 Wireframe of the TDSM using SBA (a) TDSM 1 (b) TDSM 2 50

4.1 Canvas after placing first rectangle, R1 60

4.2 Second rectangle,R2, can be placed in either region A or B from Fig. 4.1 . . . 61

4.3 Canvas is divided into three regions, I, II, and III. . . . 62

4.4 Wireframe of the TIN 63

4.5 Texture with optimal triangles packed with the packing algorithm 64

4.6 Regular texture 65

4.7 Registered output TDSM 67

4.8 Registered output TDSM 68

4.9 Registered output TDSM 69

4.10 Textured Pointcloud 70

4.11 Registered output TDSM 71

4.12 Output image canvas after packing algorithm 72

4.13 Zoomed to show details 73

5.1 illustration of superresolution. 76

5.2 Basic algorithm overview of SR using Wiener filter 79

5.3 Flowchart showing the creation of a TDSM using the optimized texel images. 80

5.4 Gaussian PSF of the imaging system, $h(x, y)$ with $\mu_x = \mu_y = 0$ and $\sigma_x = \sigma_y = 1.5$ pixels 85

5.5 Autocorrelation model for desired HR image, $r_{dd}(x, y)$ with $\sigma_d^2 = 1$ and $\rho = 0.75$ 87

5.6 3D mesh triangles 89

5.7 3D triangle projected onto all images sorted using score 90

5.8 Zoomed to show details 90

5.9 Best LR triangle in original LR image. Red pixel corresponds to all the pixel inside the triangle and blue corresponds outside the triangle within the bounding box. 91

5.10 Fig 5.9 is zoomed to show details 92

5.11 LR pixels from best triangle transformed to HR grid 93

5.12 LR pixels from all triangles transformed to HR grid 93

5.13 Zoomed to show details. The red square comprises of 3 x 3 HR pixels is an instance of an observation window 94

5.14 Observation window. Each color corresponds to a pixel from a different LR image 94

5.15 Weights based on the observation window 95

5.16 Weights based on the observation window 95

5.17 (a) Original LR triangle (b) HR triangle after applying the algorithm (c) Bilinear interpolated triangle applied to LR triangle 96

5.18 Results 1 97

5.19 Results 2 98

5.20 Results 3 99

5.21 Results 4 99

5.22 Final output TDSM for small portion 100

ACRONYMS

3D	Three-dimensional
DSM	Digital Surface Model
TDSM	Textured Digital Surface Model
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
UAV	Unmanned Aerial Vehicles
LIDAR	Light Detection and Ranging
EO	Electro-Optical
COP	Center Of Projection
FOV	Field Of View
DCM	Direction Cosine Matrix
NCC	Normalized Cross Correlation
SURF	Speeded-Up Robust Features
DEM	Digital Elevation Map
BA	Bundle Adjustment
LMA	Levenberg-Marquardt Algorithm
CBA	Conventional Bundle Adjustment
SBA	Streaming Bundle Adjustment
TIN	Triangulated Irregular Network
BFS	Breadth First Search
SR	Super-Resolution
LR	Low Resolution
HR	High Resolution
AWF	Adaptive Wiener Filter
PSF	Point Spread Function

CHAPTER 1

Introduction

Creating Three-dimensional (3D) models from real objects is a busy area of research and a growing field. These models are valuable for visualization and serve as a foundation for further analysis. Three-dimensional modeling is used in many aspects, such as cultural heritage, landscape, terrain, and even in planetary modeling. A Digital Surface Model (DSM) is an area of 3D modeling that focuses on the terrain surface. A DSM is a 3D representation of a terrain surface and is used extensively in the areas of agriculture [1], ecology [2], cultural heritage [3], forestry [4], modeling water flow [5], volcanic monitoring [6] and many others. A DSM overlaid with a texture conveys valuable information about the terrain surface and is called Textured Digital Surface Model (TDSM). Traditionally, the images are taken from full-scale aircraft with expensive positioning sensors like a Global Positioning System (GPS) and an Inertial Measurement Unit (IMU). This makes the final product very costly. With the recent development of Unmanned Aerial Vehicles (UAVs) and low-cost Micro-Electro Mechanical System (MEMS) GPS/IMU, image acquisition is much cheaper. The error in the position and attitude of the camera obtained from these low-cost sensors can be reduced with some post-processing techniques, reducing the overall cost of the final product. This research aims to create such scientifically accurate TDSM using low-cost hardware while achieving similar results obtained from a high-cost system. This chapter will cover the introduction to this study along with the research problems, research objectives, and the significance of this study.

Previously, 3D models were constructed with the help of many 2D images using the stereo vision technique [7,8]. This technique is based on how our eyes work, where the 3D points are triangulated using 2D points in stereo images. The problem with using just the photogrammetry is that the depth is not measured but inferred, making it less accurate. Another domain of data acquisition is using LIDAR (Light Detection and Ranging). It

is similar to RADAR (Radio Detection and Ranging) but uses pulsed laser light instead of a radio wave [9, 10]. The lidar source emits the laser, which strikes the target and returns back to the source. The difference in laser return times and wavelengths determines the distance between the target and the source. Such measurement can be placed as a single point in a 3D space. Scanning the object and taking a series of such measurements allows a point cloud to form. The point cloud represents the 3D shape of an object. However, texture information is absent in such a dataset. There are extensive researches to combine the digital camera and the lidar to create fused imagery and lidar data, which can truly give the 3D image of an object. Several methods have been used to create the fused lidar/imagery dataset [11–17]. Traditional approaches match features with the help of known line or point correspondences. Zang and Pless use a checkerboard pattern to match the points between the images from the camera and laser scanner [11]. Ding et al. first found the coarse camera parameters and used corner matching from the 2D image and 3D image [12]. Other methods consist of a stereo camera with Time-of-Flight(TOF) camera [13, 14], using a range sensor with an image sensor [15], utilizing reflectance image and iterative pose estimation based on robust M estimation [16], and using gradient constraints between the range intensity and a two-color image [17].

This work uses texel images for the fused lidar/imagery data. The texel camera, developed by CAIL at Utah State University [18], is comprised of both a lidar (as a depth sensor) and a digital camera (as an EO sensor). It captures both the digital image and the lidar measurement simultaneously and calibrates the information upon capture, creating a 2.5D image called a texel image. Each lidar point is mapped to an EO image point by the initial calibration, [19]. A texel camera is fitted onto the UAV and flown over the terrain for image acquisition. The camera continuously captures terrain images during the UAV flight, as shown in Fig 1.1. This simulated texel camera, shown in the green box, has 16-channel lidar (blue rays) and a digital camera (red dot). This proposed work combines these images and produces a true TDSM of the terrain. Fig 1.2 shows a simulated example of TDSM from the flight shown in Fig 1.1.

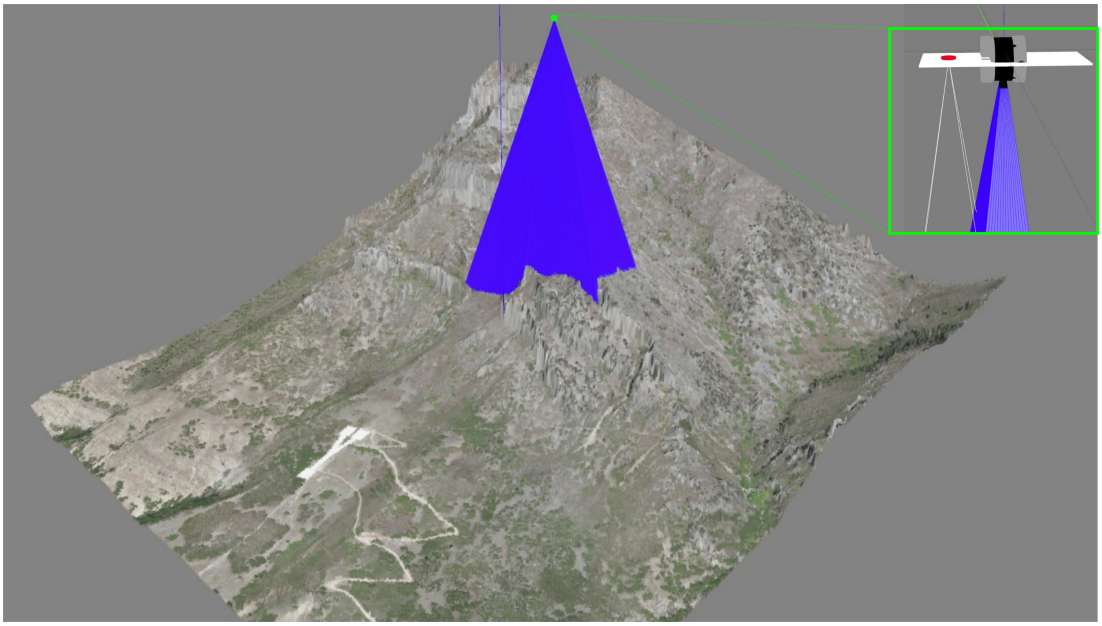


Fig. 1.1: An example flight scenario: a model of texel camera taking images of the terrain.

(a) Top View (b) Side View

Fig. 1.2: Simulated TDSM generated from the example flight above

1.1 Statement of the Problem

Before discussing the problems regarding the existing work, a brief discussion about the structure of a texel image and the texturing method to create a TDSM is necessary. As mentioned before, a texel image is called a 2.5D image since it is a 3D image but from a single point of view. Each texel image consists of a collection of 3D points. These 3D points are then triangulated using Delaunay triangulation to create a Triangulated Irregular Network (TIN), which creates a wireframe that characterizes the surface of the final TDSM. Each triangle in this wireframe is called a 3D triangle in this document. Additionally, each texel image contains a texture file, which is a digital image, and a mapping that maps each 3D point to a specific location in the digital image. The texture is overlaid on top of the wireframe surface using the mapping provided in a texel image, thus creating a TDSM. A single texel image can be thought of as a subset of a TDSM. The registration of these multiple texel images allows us to create a TDSM of a large area.

Creating a 3D model using texel images has shown improvement over the years. The limitation of using only two texel images was extended up to multiple texel images [20]. Bybee and Budge used texel swath and introduced the concept of decreasing the error in projection points instead of the actual position of 3D points [21]. The cost function reduced by bundle adjustment depends on the projection of a 3D point onto an image and an actual range measurement. Bundle adjustment is an optimization algorithm that gets more expensive in computation and memory as the number of texel images increases. This limits the number of texel images used to create a TDSM. A typical image acquisition flight of 10 minutes consists of well over 3000 texel images. This presents a problem as the computer used to generate TDSM cannot handle that extensive data. As a result, the existing research is inadequate for any practical application when the area is large.

Another problem in existing research is the texturing of the final TDSM. During registration, all 3D points are registered together in a common space and triangulated to create a TIN surface. To texture this surface, the texture from each texel image can be used. In the existing work, final texture was selected based on the nearest 3D points. The texture

is selected from the texel image containing the DSM's ortho-view for a single-channel lidar. This point-based texturing approach is problematic because the final TDSM always has the top-view texture, which may not be suitable for texturing the sides of an object when the terrain has an elevation change. Additionally, for a texel camera with a multiple-channel lidar, this texturing method can cause distortion in the final texture even in flat areas, as 3D points from different texel images may lie close together.

For example, consider a 3D triangle from a TIN shown in Fig 1.3, denoted as a yellow-colored triangle. This 3D triangle is located on a hill that needs to be textured and has six candidate triangles. Each of these triangle is a part of different images. Since this yellow-colored triangle can be seen in all of these six images, it could be textured from any one of them. However, the quality of the TDSM depends on which texture is selected. The existing method will use Triangle 3 to texture the yellow triangle for a single channel lidar or could be textured using a texture from multiple different triangles for a texel camera with multiple channel lidar. Nevertheless, as seen in Fig 1.3, Triangle 5 seems to be a good match for texturing it.

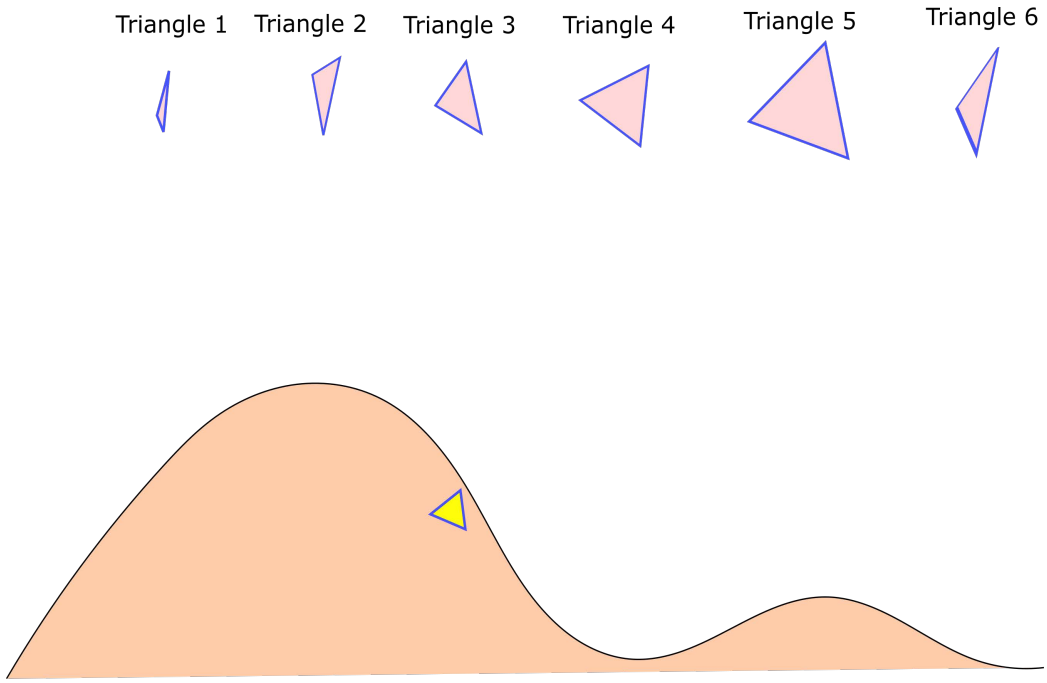


Fig. 1.3: 3D triangle (marked with yellow) with six different candidate images

1.2 Research aims, objectives, and research question

Given the absence of handling a large number of texel images to create TDSM in the existing work, this study will aim to utilize a streaming method to optimize the chunks of texel images instead of optimizing all available texel images. A sliding window is implemented, and the optimization is carried out in this window, thus able to create TDSM from a large number of texel images. This sub-optimal optimization is compared with the optimal method of registering all available texel images together. This research also aims to improve the final texture of the TDSM. By switching from point-based texturing to triangle-based texturing, the triangulated network formed using point clouds is textured using a texture triangle. The selection of the best texture triangle based on various criteria aims to improve the texture of the TDSM compared to the existing point-based texturization. An algorithm for packing such triangles is designed that aims to reduce the size of the image file. Finally, this study seeks to improve the output texture by applying superresolution. The availability of texture triangles from different views gives rise to sub-pixel displacements. A wiener filter is designed based on these displacements, which allows superresolution to be successful.

The research objectives for this study are listed below:

1. **RO1**: To implement Streaming Bundle Adjustment needed to handle a large number of texel images.

2. **RO2**: To formulate a new cost function for the streaming method and evaluate its effectiveness on the final result.

3. **RO3**: To develop an algorithm for the selection of the best triangle texture using various criteria and evaluate its effectiveness.

4. **RO4**: To formulate a packing algorithm for packing the best triangle into an image canvas.

5. **RO5**: To use a superresolution technique on the 2D image to improve the output quality.

The research questions that this work will seek to answer are given below.

1. **RQ1**: How effective is Streaming Bundle Adjustment compared to Conventional Bundle Adjustment?
2. **RQ2**: What are the advantages and disadvantages of choosing optimal texture fragments?
3. **RQ3**: How effective is the texture packing algorithm in reducing the image size?
4. **RQ4**: How much can the texture of the TDSM can be improved using super-resolution?

1.3 Significance of the study and limitations

The demand for scientifically accurate terrain maps of the earth is currently high. These maps are highly desirable in the field of agriculture, including mapping the topography of areas, crop foliage density and growth, and determining canal structure to automate water delivery systems. Typical traditional surveys and manned aircraft data collection are expensive for small areas. This study will contribute to the agricultural and scientific field by producing a quality TDSM at a fraction of the cost compared to the existing system. Improving the texture using this work adds a valuable contribution to the quality of TDSM. Because of the streaming nature of the algorithm, this study can be applied to large-scale applications like highway design and large terrain maps. This will provide real-world value in the field of transportation and remote sensing.

In this work, it is assumed that all the lidar points are correct to a degree and no bad lidar points are thrown out. The other limitation of this study is that the texel images used as inputs to the algorithm are taken from the flight pattern that is only moving forward. There is no consideration for the overlap in the area that might occur for a U-flight pattern.

1.4 Outline

Chapter 1 has provided a brief introduction and context to the research work. A short background work has also been discussed. The objectives and the research questions have been identified, along with the significance of the study. The limitations of the study were also discussed.

Chapter 2 provides a concept of the texel camera and the basics of camera geometry and image processing. It also provides some background on the topics needed for this work.

Chapter 3 presents the background and methodology related to Streaming Bundle Adjustment. A new cost function is formulated that will be used in the optimization process. The result using the algorithm is shown along with the discussion.

Chapter 4 talks about the selection of optimal texture for the 3D structure. Various criteria for the best triangle texture are discussed, and the algorithm to pack these triangles is presented.

The theory and the algorithm related to super-resolution are discussed in Chapter 5. An algorithm to create a high-resolution texture triangle from the available LR texture triangle is presented along with the improved result.

Chapters 3, 4, and 5 have individual results and discussions for each topic. Finally, Chapter 6 presents the conclusion for this study and discusses suggested future works.

CHAPTER 2

Texel Camera and Fundamental definition

A texel camera is an instrument developed in The Center for Advanced Imaging Ladar (CAIL) at Utah State University, which comprises a range sensor and an Electro-Optical (EO) sensor mounted together in the same mechanical frame. A lidar camera is used as the range sensor, and a digital camera is used as the EO sensor. The EO sensor captures the digital image of the scene, whereas the lidar sensor captures the range information. The camera fuses the 2D digital image and 3D depth image to create a 2.5D image, called a texel image. It is termed 2.5D instead of a 3D image because it is a 3D image from a single point of view.

2.1 Coboresighted Texel Camera

In the coboresighted texel camera, the lidar sensor and EO camera are arranged so that the center of projection (COP) of both the lidar and the EO sensor fall at the same point in space. A cold mirror is placed between them to achieve the common center of projection. This cold mirror has a unique property that allows it to reflect light below about 760 nm wavelength. The lidar sensor of the texel camera operates at a wavelength of 870 nm, in the near-infrared (NIR) range. In contrast, the EO camera operates at standard optical wavelengths (approximately 400 to 700 nm). Thus, the image captured by the digital camera is reflected through the mirror, whereas the lidar data captured by the depth camera passes through this cold mirror. The tilt of this cold mirror is determined during the calibration process and held by a mechanical mount along with both sensors. This allows two sensors to be coboresighted throughout the image acquisition process [19].

Fig 2.1 shows the ray diagram of a coboresighted texel camera showing the fundamental function of the cold mirror. In the coboresighted camera, the Field Of View (FOV) of both the depth camera and the EO camera are the same. Fig 2.2 shows the second generation

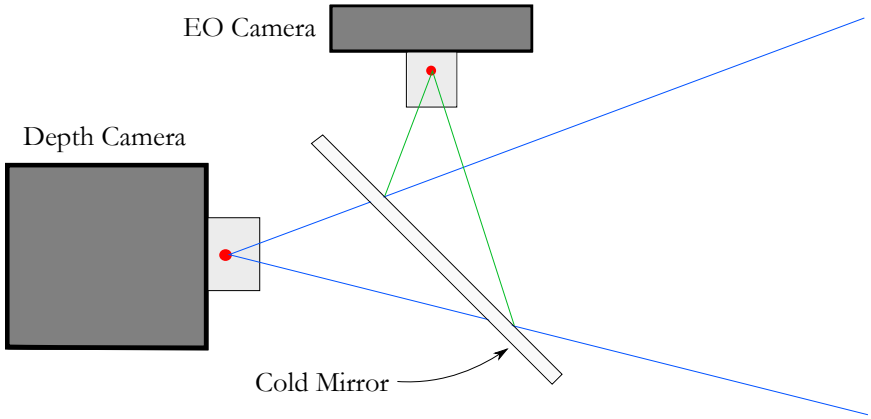


Fig. 2.1: Ray diagram of a coboresighted texel camera.

texel camera.

2.2 Bistatic Texel Camera

CAIL, along with the collaboration of Aggie Air, developed a texel camera that consists of a rotational lidar sensor, a digital camera, an Inertial Measurement Unit (IMU), and an onboard processor. Unlike its predecessor, the lidar sensor and EO sensor are placed side by side, and no cold mirror is used. This bistatic nature of the sensor package means the two sensors will not be coboresighted. The COP of both cameras is at a different physical location which cause them to have different FOVs and add further complexity to the calibration of the texel camera. Fig 2.3 shows the model diagram of a bistatic texel camera where it can be seen that the FOV is different for each camera. Fig 2.4 shows the texel camera developed by CAIL. It contains a 16-channel lidar camera, a digital camera, an IMU, and an onboard processor. An SD card is used to save the captured images during the flight.

2.3 Texel Image

The data produced by the texel camera is called the texel image. The fundamental data structure of a texel image is that it contains a digital image and a collection of data containing the location of each 3D points with range measurement and a mapping between the 3D points and digital image called UV mapping. Generally, the resolution of the lidar

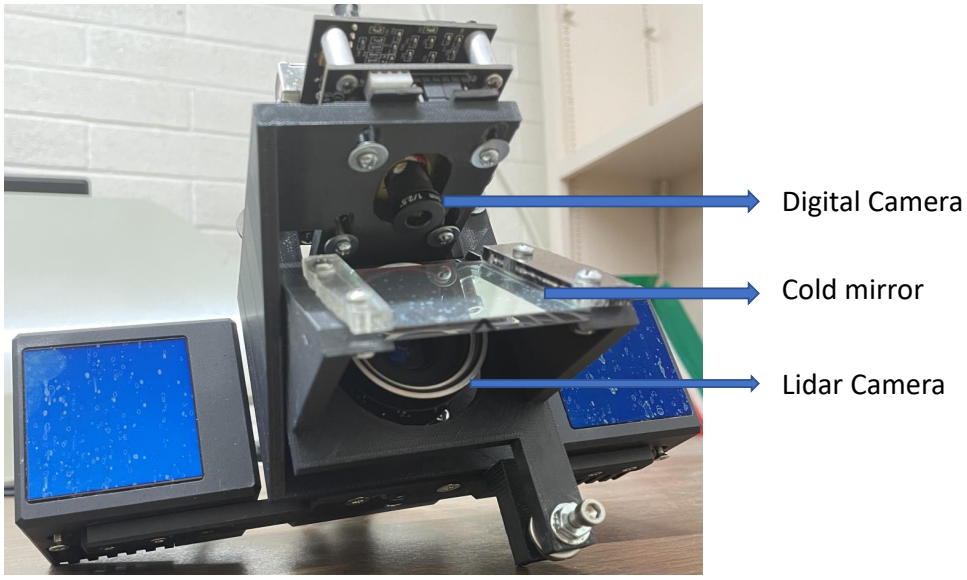


Fig. 2.2: Second generation coboresighted texel camera.

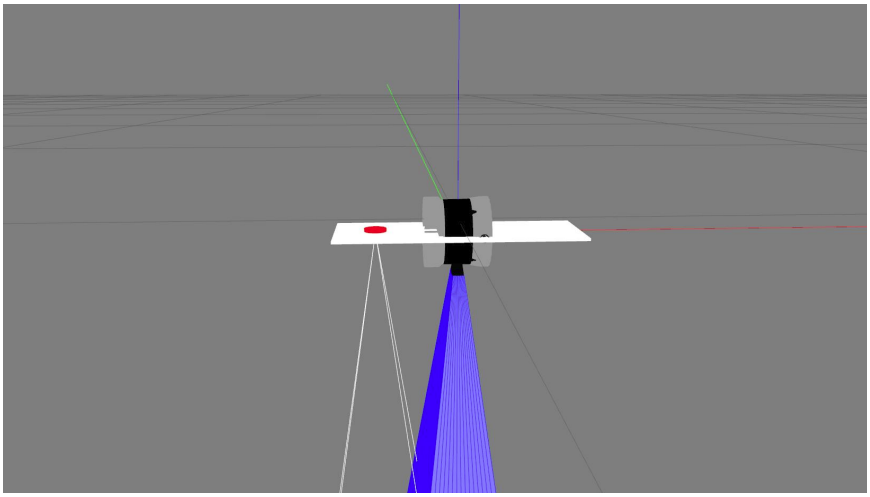


Fig. 2.3: Bistatic texel camera model.

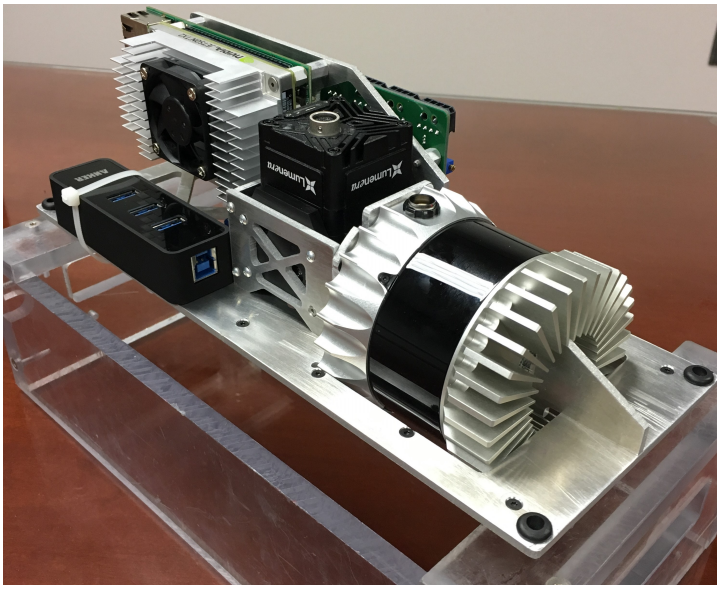


Fig. 2.4: Bistatic texel camera developed by CAIL.

camera is far lower than the EO camera, and the UV mapping assigned each lidar measurement to the calibrated location on the EO image. The lidar measurement can have other attributes like intensity, time, row and column number, etc.

2.4 Quaternion

A quaternion is a 4-tuple number system that is an extension to the complex number and lives in a 4-dimensional space. It consists of three imaginary dimensions perpendicular to each other, describing the space and a real number perpendicular to this space in the fourth dimension. The three imaginary dimensions are denoted as (i, j, k). Legend has it that W. R. Hamilton, an Irish mathematician, had the idea of a quaternion equation while walking and carved this famous equation $i^2 = j^2 = k^2 = ijk$ on the stone of the bridge over a canal [22].

The complex number can be used to describe rotation in 2D. Similarly, the quaternion can be used in describing rotation in 3D and thus has applications in computer graphics, robotics, and any field involving 3D orientation. The well-known Euler angles and 3 x 3 Direction Cosine Matrix (DCM) are also used in describing rotation in 3D. However, the

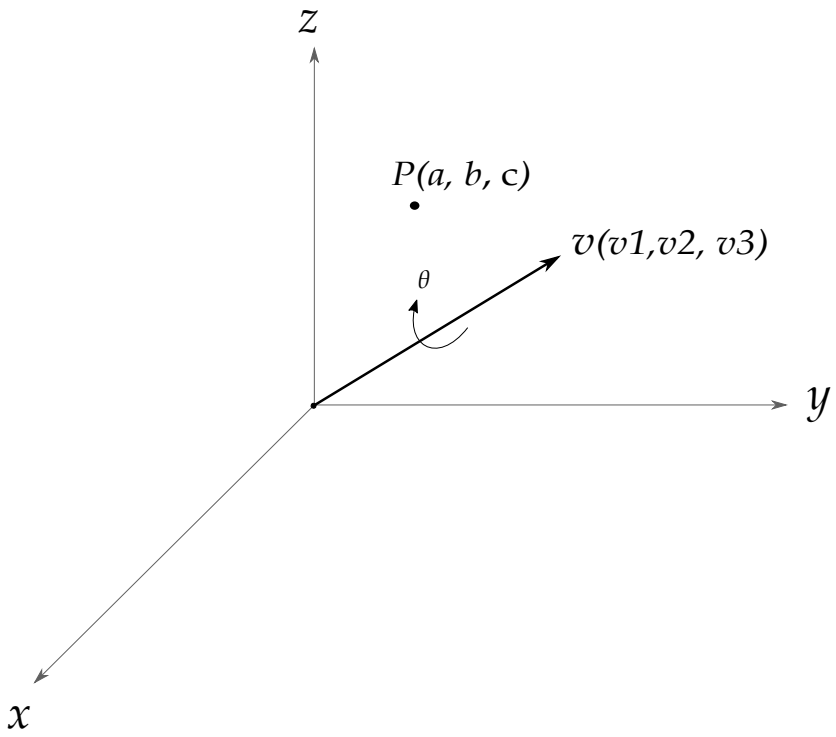


Fig. 2.5: Rotation representation.

rotation using quaternions is computationally more efficient and avoids many numerical errors arising in these other methods. It also avoids the gimbal lock that may arise while using Euler angles [23].

Let us consider a 3D space with the basis vector $\boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$. A point $\mathbf{P}$ can be represented as,

$$\mathbf{P} = a\boldsymbol{i} + b\boldsymbol{j} + c\boldsymbol{k}.$$

Let $\boldsymbol{v} = [v1, v2, v3]$ be the vector on this space and θ be the angle which rotates point P around the axis $\boldsymbol{v}$ as shown in Fig 2.5, then the quaternion that define this rotation is

given by $\mathbf{q} = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}$ where,

$$\begin{aligned} q_0 &= \cos\left(\tfrac{\theta}{2}\right) \\ q_1 &= v1\sin\left(\tfrac{\theta}{2}\right) \\ q_2 &= v2\sin\left(\tfrac{\theta}{2}\right) \\ q_3 &= v3\sin\left(\tfrac{\theta}{2}\right). \end{aligned} \tag{2.1}$$

The final point $\mathbf{P}^{'}$ after rotation is given by,

$$\mathbf{P}^{'} = q\mathbf{P}q^{-1},$$

where $\mathbf{q}^{-1} = q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k}$.

In this work, DCM is used to rotate points from one coordinate system to another. Even though the DCM matrix has nine elements, it only has three degrees of freedom. This makes it harder to solve the optimization problem because of the six non-linear constraints to enforce orthonormality ($R^TR = I$) and the additional constraint $det(R) = +1$. With the quaternion, there is only one extra constraint: the rotation has to be a unit quaternion, which is relatively easy to deal with. This makes it possible to find a closed form solution to the optimization problem, which is why quaternions are used in solving final bundle adjustment optimization in this work.

The conversion between DCM and quaternion is straightforward and is given below. For any unit quaternion $\mathbf{q}$, the equivalent DCM matrix R is given as:

$$[R] = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & (q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}, \tag{2.2}$$

There are different ways to convert the DCM matrix to quaternion. Consider a DCM matrix R with elements R_{ij}, where i and j are row and column values respectively. Because

the rotation is a unit quaternion, we have

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \tag{2.3}$$

From (2.2) and (2.3), it is straightforward to show that:

$$\begin{aligned} q_0^2 &= \frac{1}{4}(1 + R_{11} + R_{22} + R_{33}), \\ q_1^2 &= \frac{1}{4}(1 + R_{11} - R_{22} - R_{33}), \\ q_2^2 &= \frac{1}{4}(1 - R_{11} + R_{22} - R_{33}), \\ q_3^2 &= \frac{1}{4}(1 - R_{11} - R_{22} + R_{33}). \end{aligned} \tag{2.4}$$

Taking the square root of the maximum value amongst those terms provides the particular value for that term. The remaining tuple value can be computed using the appropriate equation from (2.5).

$$\mathbf{q} = \frac{1}{4q_0}\begin{bmatrix} 4q_0^2 \\ R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} = \frac{1}{4q_1}\begin{bmatrix} R_{32} - R_{23} \\ 4q_1^2 \\ R_{12} + R_{21} \\ R_{13} + R_{31} \end{bmatrix} = \frac{1}{4q_2}\begin{bmatrix} R_{13} - R_{31} \\ R_{12} + R_{21} \\ 4q_2^2 \\ R_{23} - R_{32} \end{bmatrix} = \frac{1}{4q_3}\begin{bmatrix} R_{21} - R_{12} \\ R_{13} + R_{31} \\ R_{23} - R_{32} \\ 4q_3^2 \end{bmatrix}. \tag{2.5}$$

2.5 Camera pose and transformation between coordinate frames

The IMU unit attached to the texel camera gives a rough estimate of the camera's pose during the time of capture of each texel image. Initially, the IMU gives the pose in the North-East-Down (NED) frame, and the image acquisition software transforms this pose into an image-coordinate system and records it as the pose of the texel image. The x, y, z axes in the image-coordinate system follow the convention where x is the cross-track direction, y is the in-track direction of the camera, and z is the up direction. All the 3D points recorded in the texel image are defined in this image-coordinate system. Each camera's pose consists of attitude and translation for each image frame relative to some common frame, called a world-coordinate system. This pose can be written in matrix form

as $[R|\mathbf{t}]$ where R is the 3x3 rotation matrix and $\mathbf{t}$ is the 3x1 translation vector. The first measurement recorded from the INS during acquisition is set as the origin for the world coordinate system and is written in matrix form as

$$[R_{\mathcal{O}}|\mathbf{t}_{\mathcal{O}}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.6}$$

For any j^{th} camera, the pose matrix in terms of a quaternion can be represented as

$$[R_j|\mathbf{t}_j] = \begin{bmatrix} 1-\frac{2(q_{j2}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j2}-q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j3}+q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & t_{jx} \\ \frac{2(q_{j1}q_{j2}+q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1-\frac{2(q_{j1}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}-q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & t_{jy} \\ \frac{2(q_{j1}q_{j3}-q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}+q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1-\frac{2(q_{j2}^2+q_{j1}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & t_{jz} \end{bmatrix}, \tag{2.7}$$

where $[q_{j0}, q_{j1}, q_{j2}, q_{j3}]$ is the quaternion representation of the rotation and $[t_{jx}, t_{jy}, t_{jz}]$ is the location of j^{th} camera center in the world coordinate system [24].

Using the pose defined in (2.7), it is easy to transform any point from the world coordinate frame to the image coordinate frame and vice versa. Let $\chi_j = [\chi_{j_x}, \chi_{j_y}, \chi_{j_z}]^T$ be the 3D point defined in j^{th} camera frame and $\chi\mathcal{O} = [\chi\mathcal{O}_x, \chi\mathcal{O}_y, \chi\mathcal{O}_z]^T$ is the same point in the world coordinate frame. To transform a point from a world coordinate frame to an image-coordinate frame, we have

$$\begin{bmatrix} \chi_{j_x} \\ \chi_{j_y} \\ \chi_{j_z} \end{bmatrix} = \begin{bmatrix} 1-\frac{2(q_{j2}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j2}-q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j3}+q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \\ \frac{2(q_{j1}q_{j2}+q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1-\frac{2(q_{j1}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}-q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \\ \frac{2(q_{j1}q_{j3}-q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}+q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1-\frac{2(q_{j2}^2+q_{j1}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \end{bmatrix}^T \begin{bmatrix} \chi\mathcal{O}_x - t_{jx} \\ \chi\mathcal{O}_y - t_{jy} \\ \chi\mathcal{O}_z - t_{jz} \end{bmatrix}. \tag{2.8}$$

Similarly, to transform a point defined in the image coordinate frame to the world coordinate frame, we have

$$\begin{bmatrix} \chi_{\mathcal{O}_x} \\ \chi_{\mathcal{O}_y} \\ \chi_{\mathcal{O}_z} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 - \frac{2(q_{j2}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j2}-q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j3}+q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & t_{jx} \\ \frac{2(q_{j1}q_{j2}+q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1 - \frac{2(q_{j1}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}-q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & t_{jy} \\ \frac{2(q_{j1}q_{j3}-q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}+q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1 - \frac{2(q_{j2}^2+q_{j1}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & t_{jz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \chi_{j_x} \\ \chi_{j_y} \\ \chi_{j_z} \\ 1 \end{bmatrix}. \tag{2.9}$$

2.6 Normalized plane

It was assumed that both the lidar camera and EO camera follow the pinhole camera model. If we assume a pinhole camera model, then the COP of the camera lies at the pinhole, and the image plane lies behind the camera. This phenomenon is similar to the way the human eye works. The light passes into the eye through the pupil to project an upside-down image of the real world on the back of the retina. The brain then interprets this image by flipping it so that we experience the real world with the right side up. Similarly, the light passes through the pinhole camera and projects an upside-down image on the image plane. To simplify, we assume the image plane is in front of the camera. If this image plane is kept at a unit distance perpendicular to the look direction, it is called a normalized camera plane. In this work, the normalized camera plane is placed at $z = 1$ meter away from the center of the camera along the principal axis.

In the case of a coboresighted texel camera, both the sensors have the same FOV and hence a common normalized plane. This makes transformation between lidar frame and camera frame easy as one can project onto a normalized plane and back-project onto other sensor frames. However, in the bistatic texel camera, due to the physical separation between the two sensors, the FOV is different for each sensor. This results in having two different normalized planes, one for each sensor. In this document, the normalized plane for the depth sensor is called the normalized lidar plane, and the normalized plane for the EO sensor is called the normalized camera plane. The normalized lidar plane is also placed at $z = 1$ meter away from the center of the lidar camera along the principle axis.

In this work, since the final optimization is done on the normalized camera plane, a mapping from the normalized lidar plane to the normalized camera plane is needed. Such non-linear mapping is defined by a second-order polynomial function, where the coefficients are found during the calibration stage of the sensor, and the camera calibration matrix, K [19].

2.7 Projection and range coordinate system

Since the true range of the object is measured using the lidar sensor, it is desired to change the optimization problem by finding the projection and the range of a 3D point. When an object is projected onto the normalized plane, its depth information is lost. But, with the help of the range measurement, the position of a 3D point can be obtained. This mapping from the position of a 3D point to its projection-range coordinate is unique and one-to-one (except when the z-coordinate of a 3D point is 0). In the case of a bistatic texel camera, this projection is done onto the normalized lidar plane. If $[\chi_{j_x}, \chi_{j_y}, \chi_{j_z}]$ is the 3D point in an image coordinate frame and $\tilde{\boldsymbol{X}}_j$ is the projection-range representation of the same 3D point, then the mapping from Cartesian coordinates to projection-range coordinates is given as

$$\tilde{\boldsymbol{X}}_j = \begin{bmatrix} \tilde{x}_j \\ \tilde{y}_j \\ \lambda_j \end{bmatrix} = \begin{bmatrix} \dfrac{\chi_{j_x}}{\chi_{j_z}} \\ \dfrac{\chi_{j_y}}{\chi_{j_z}} \\ \sqrt{\chi_{j_x}^2 + \chi_{j_y}^2 + \chi_{j_z}^2} \end{bmatrix}. \tag{2.10}$$

To convert back to the Cartesian coordinate, we have

$$\begin{bmatrix} \chi_{j_x} \\ \chi_{j_y} \\ \chi_{j_z} \end{bmatrix} = \begin{bmatrix} \tilde{x}_j \dfrac{\lambda_j}{\sqrt{\tilde{x}_j^2 + \tilde{y}_j^2 + 1}} \\ \tilde{y}_j \dfrac{\lambda_j}{\sqrt{\tilde{x}_j^2 + \tilde{y}_j^2 + 1}} \\ \dfrac{\lambda_j}{\sqrt{\tilde{x}_j^2 + \tilde{y}_j^2 + 1}} \end{bmatrix}. \tag{2.11}$$

2.8 Calibration of bistatic sensor and parallax correction

Before the data is acquired from the texel camera, a calibration must be carried out. The calibration is carried out in mainly three steps.

- Calibration of Digital Camera
- Lidar and Digital camera alignment
- Estimation of mapping from Lidar to Camera

The digital camera is first calibrated to remove any barrel and pincushion distortion introduced by the camera. Heikkilä and Silvén [25] proposed the theory behind this calibration method and was implemented in the Camera calibration toolbox in MATLAB [26]. This toolbox estimates the intrinsic camera calibration matrix (also called a camera calibration matrix), K, and is given as

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.12}$$

where f_x and f_y are focal length in terms of pixel dimensions in the x and y direction respectively, s represents the skew of the x and y axes, and (c_x, c_y) represents the principal point in pixel dimensions. We assumed the x and y axes to be perpendicular to each other and set the skew parameter to zero. In this work, the camera matrix is estimated as

$$K = \begin{bmatrix} 5301.25619228 & 0 & 2173.70289326 \\ 0 & -5306.18965153 & 1467.62014742 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.13}$$

With the help of K, a point in the normalized image plane can be mapped to a pixel value and vice versa. Any point $\boldsymbol{x} = [x_j, y_j, 1]$ in the normalized image plane can be mapped to a pixel value $\boldsymbol{p} = [c, r, 1]$ by multiplying with K as $\boldsymbol{p} = K\boldsymbol{x}$. Similarly, the pixel value can be mapped to a point in the normalized image plane by multiplying with K^{-1} as $\boldsymbol{x} = K^{-1}\boldsymbol{p}$.

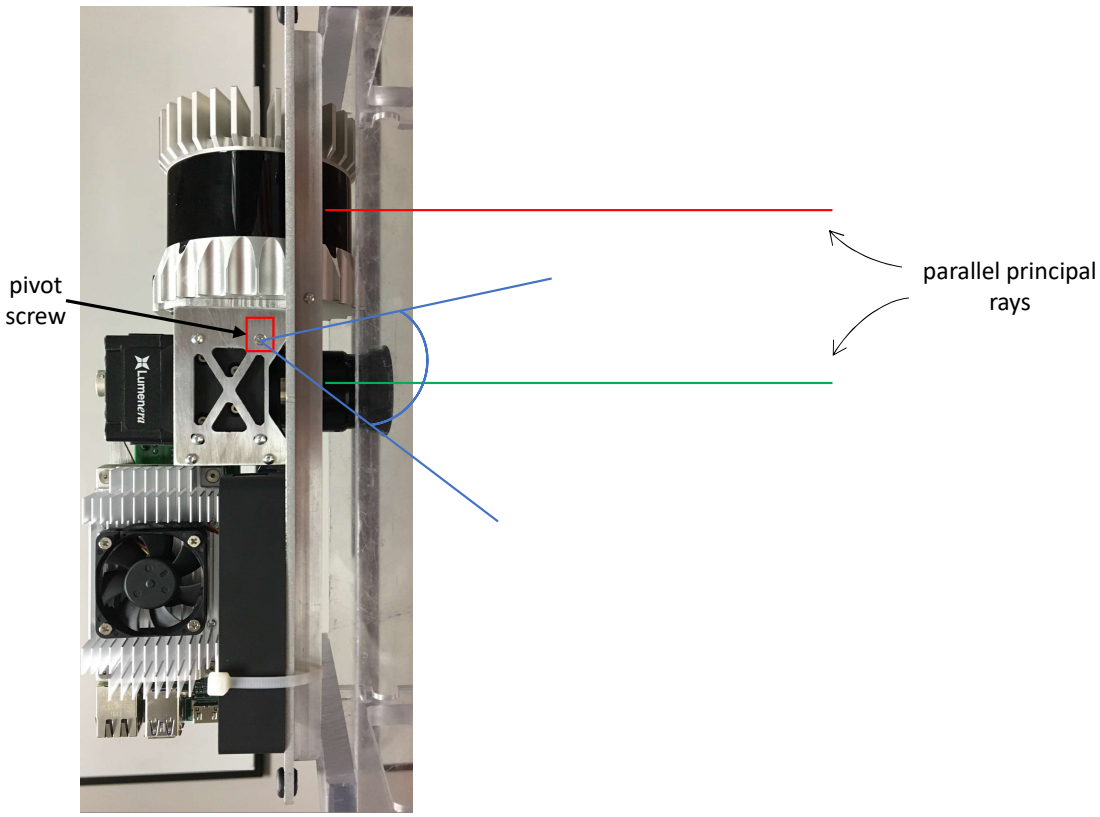


Fig. 2.6: Aligninment of both sensors such that their principal rays are parallel.

The next step is to align the two sensors such that the principal rays of the camera and the lidar are parallel to each other. The physical alignment is done with the help of a pivot screw as shown in Fig 2.6. In the coboresighted texel camera, the cold mirror is tilted such that both principal rays coincide, thus removing the concern for parallax. Due to

the physical separation of the two sensors, parallax is always present in the bistatic sensor and needs to be compensated. This is done by shifting the normalized lidar points in the in-track (y) direction. If the two sensors are aligned, then the parallax depends only on the z-distance from the camera to the object. Since the physical separation of the sensors only occurs in the y-direction, the parallax compensation is done only in the y-coordinates in the normalized lidar plane. If d_0 is the distance between the image sensor and the lidar sensor, z_{map} is the mapping distance i.e. distance from the texel camera to the calibration plane during mapping, then the parallax offset, Δy, that must be applied for a particular z is given as [27]

$$\begin{aligned} \Delta y_n &= \left(y_n + \frac{d_0}{z_{map}}\right) - \left(y_n + \frac{d_0}{z}\right) \\ \Delta y_n &= d_0\left(\frac{1}{z_{map}} - \frac{1}{z}\right). \end{aligned} \tag{2.14}$$

From (2.14), it can be seen that the parallax would be zero for an object lying at the same distance as the mapping distance.

Lastly, the mapping between the lidar shots and the image pixels is estimated. This is done by placing the texel camera on a tripod, taking pictures, and finding N-correspondences between lidar shots and pixel coordinates. The model that relates the point in the normalized lidar plane and the row-column of the digital camera is given as [19]

$$\begin{aligned} c &= g_1 + g_2\tilde{x}_n + g_3\tilde{x}_n^2 + g_4\tilde{x}_n^3 + g_5\tilde{y}_n + g_6\tilde{x}_n\tilde{y}_n + g_7\tilde{y}_n^2 + g_8\tilde{x}_n\tilde{y}_n^2 + g_9\tilde{x}_n^5 + g_{10}\tilde{x}_n\tilde{y}_n^4 + g_{11}\tilde{x}_n^3\tilde{y}_n^2 \\ r &= h_1 + h_2\tilde{y}_n + h_3\tilde{y}_n^2 + h_4\tilde{y}_n^3 + h_5\tilde{x}_n + h_6\tilde{y}_n\tilde{x}_n + h_7\tilde{x}_n^2 + h_8\tilde{y}_n\tilde{x}_n^2 + h_9\tilde{y}_n^5 + h_{10}\tilde{y}_n\tilde{x}_n^4 + h_{11}\tilde{y}_n^3\tilde{x}_n^2 \end{aligned} \tag{2.15}$$

Using these $\{(c,r)_i\} \leftrightarrow \{(x_n,y_n)_i\}$ correspondence pairs, all 22 calibration parameters are found using a maximum likelihood solution.

2.9 Formation of the Projection matrix

The first step in the registration is to find the common projection points of a 3D point between all available texel images. In a texel image, there is a 2D point in the digital image that corresponds with each 3D lidar point. This mapping is created during the calibration of the texel camera. It consists of the (u, v) coordinates in the 2D digital image that are the projection of the 3D lidar points on the image, creating a (u, v, x, y, z) texel image correspondence. Since the lidar data is of lower resolution than the 2D image, any image pixels that are not a direct projection of a 3D lidar point do not have a measured corresponding 3D point.

A projection matrix is a table that consists of image projection for each 3D point onto all available texel images. Given two texel images, j and k, the 3D point $\mathbf{b}_i$ captured by image j can be seen in both images. However, calibrated image projection and a measured range value for $\mathbf{b}_i$ are available only for image j and can be entered in the projection matrix. To fill the entry for projection of $\mathbf{b}_i$ onto image k, we have to find the image projection using correlation. Since there is no measured range, the range value is always zero in this case. Table 2.1 shows an example of the projection matrix where $\mathbf{a}_j$ is the pose of the j^{th} camera, I_j, and $\mathbf{b}_i$ is the 3D points such that

$$\mathbf{b}_1 \in I_1,\ \mathbf{b}_2 \in I_2,\ \mathbf{b}_3, \mathbf{b}_4 \in I_3,\ \mathbf{b}_5 \in I_4,\ \mathbf{b}_6 \in I_5$$

The entries in the projection matrix where the range values λ_{ij} are available are the projection and range obtained from calibration and measured with the lidar sensor, respectively. To find other entries using correlation, a small patch around the calibrated projection point is compared with a similar patch on other images by searching an entire image to find the correspondence. This is a costly process in terms of computation. Homography is used to reduce this search size. With the help of homography, a candidate correspondence is found, and the search area can be limited around this candidate correspondence. An exact match is found using Normalized Cross Correlation (NCC). Texel images are captured one after another very quickly; thus, EO imagery does not change significantly from image to image. This enables NCC to work efficiently.

Table 2.1: An example of a projection matrix

	a_1	a_2	a_3	a_4	a_5
b_1	$\boldsymbol{X_{11}}$ $[x_{11}, y_{11}, \lambda_{11}]$	$\boldsymbol{X_{12}}$ $[x_{12}, y_{12}, 0]$	$\boldsymbol{X_{13}}$ $[x_{13}, y_{13}, 0]$	$\boldsymbol{X_{14}}$ $[x_{14}, y_{14}, 0]$	$\boldsymbol{X_{15}}$ $[x_{15}, y_{15}, 0]$
b_2	$\boldsymbol{X_{21}}$ $[x_{21}, y_{21}, 0]$	$\boldsymbol{X_{22}}$ $[x_{22}, y_{22}, \lambda_{22}]$	$\boldsymbol{X_{23}}$ $[x_{23}, y_{23}, 0]$	$\boldsymbol{X_{24}}$ $[x_{24}, y_{24}, 0]$	$\boldsymbol{X_{25}}$ $[x_{25}, y_{25}, 0]$
b_3	$\boldsymbol{X_{31}}$ $[x_{31}, y_{31}, 0]$	$\boldsymbol{X_{32}}$ $[x_{32}, y_{32}, 0]$	$\boldsymbol{X_{33}}$ $[x_{33}, y_{33}, \lambda_{33}]$	$\boldsymbol{X_{34}}$ $[x_{34}, y_{34}, 0]$	$\boldsymbol{X_{35}}$ $[x_{35}, y_{35}, 0]$
b_4	$\boldsymbol{X_{41}}$ $[x_{41}, y_{41}, 0]$	$\boldsymbol{X_{42}}$ $[x_{42}, y_{42}, 0]$	$\boldsymbol{X_{43}}$ $[x_{43}, y_{43}, \lambda_{43}]$	$\boldsymbol{X_{44}}$ $[x_{44}, y_{44}, 0]$	$\boldsymbol{X_{45}}$ $[x_{45}, y_{45}, 0]$
b_5	$\boldsymbol{X_{51}}$ $[x_{51}, y_{51}, 0]$	$\boldsymbol{X_{52}}$ $[x_{52}, y_{52}, 0]$	$\boldsymbol{X_{53}}$ $[x_{53}, y_{53}, 0]$	$\boldsymbol{X_{54}}$ $[x_{54}, y_{54}, \lambda_{54}]$	$\boldsymbol{X_{55}}$ $[x_{55}, y_{55}, 0]$
b_6	$\boldsymbol{X_{61}}$ $[x_{61}, y_{61}, 0]$	$\boldsymbol{X_{62}}$ $[x_{62}, y_{62}, 0]$	$\boldsymbol{X_{63}}$ $[x_{63}, y_{63}, 0]$	$\boldsymbol{X_{64}}$ $[x_{64}, y_{64}, 0]$	$\boldsymbol{X_{65}}$ $[x_{65}, y_{65}, \lambda_{65}]$

Features from each image are computed to find the homography between two adjacent images. This work uses Speeded-Up Robust Features (SURF) as a feature detector and descriptor [28]. SURF is an innovative method used in computer vision and image processing applications to detect and match features in an image. SURF approximates the Hessian matrix in order to find keypoints. An orientation histogram then describes these keypoints. Since SURF is scale and rotation-invariant, it is fast, robust, and accurate compared to other feature extraction algorithms. Homography is computed using the least square method using all matched features between two images. To find the projection points of the 3D points of one image on its non-adjacent images, the homographies computed from each pair are cascaded to find the estimate of the homography between those images. Using these computed homographies, all other entries of the projection matrix are filled. This projection matrix is used in the final optimization.

CHAPTER 3

Streaming Bundle Adjustment

3.1 Introduction

A Textured Digital Surface Map (TDSM) is a 3D representation of the terrain surface with a textured overlay. Due to the usefulness of TDSM in various areas, creating accurate TDSMs using low-cost instruments is gaining interest in scientific and research fields. TDSMs created using only photography do not give the scale information about the final TDSM. Even with ground control points [29], the measured 3D information using lidar is much more accurate than the inferred 3D information. Moreover, lidar has the added functionality to pierce through the foliage to hit the ground and can be used to create a Digital Elevation Map (DEM).

The Inertial Navigation System (INS) in the texel camera, which consists of a GPS receiver and an IMU, measures the current position and attitude of the lidar sensor when the laser pulse is transmitted. The 3D location of the object can be transformed into a world coordinate system using this information, together with the measured range and lidar position. A point cloud is created as the lidar directs laser shots to different locations. The point cloud is a collection of 3D points that describes the scanned object's surface. As the lidar sends a series of shots, the digital camera captures an image of the scanned object. The image capture is synchronized with the lidar scanner so that the pixels corresponding to the location of the lidar shots are known [30]. To create accurate point clouds, attitude and position information when the lidar shots are taken are needed with high accuracy. The low-cost INS cannot provide such information with the accuracy needed as it only provides coarse measurements. Bundle adjustment(BA) is applied to optimize the point clouds and pose by exploiting both lidar and digital image data. A triangulated network is created from these optimized point clouds, and the texture obtained from the digital camera is

overlaid on top, creating a TDSM.

The use of texel images has shown an advantage in the past works [31, 32]. One challenge in the past is the application in large-scale TDSM construction. BA, the heart of the algorithm, is a bottleneck when the dataset grows large. BA is a non-linear least square optimization problem over the 3D structure and viewing parameters and is usually solved with the Levenberg-Marquardt (LM) algorithm. The LM algorithm has proven to be efficient and easy to implement and is usually chosen to solve the BA problem. The standard LM algorithm has a time complexity of $\mathcal{O}((m+n)^3)$ and storage complexity of $\mathcal{O}((mn(m+n))$, where m and n are the total number of images and 3D points, respectively. Because of the geometry of the multi-view reconstruction problem, the system tends to be sparse and can be exploited. However, even with the use of a sparse solver, BA is very slow and memory inefficient, and often the traditional BA is not feasible for large-scale problems [33]. As a result, many algorithms have been proposed to solve large-scale BA problems. These methods are divided into two main groups. The first is to solve the BA problem as efficiently as possible, while the second focuses on reducing BA size. With the use of sparsity of the matrix and using the Schur complement approach, researchers have reduced the computation to $\mathcal{O}(m^3 + mn)$ and memory use to $\mathcal{O}(mn)$ [7]. The efficiency in the computational step is done using the secondary sparseness in the camera system and has been shown to outperform regular sparse BA even with a dense dataset [34]. Other works have used conjugate gradients and novel preconditioning to replace the Cholesky factorization [35]. Conjugate gradient done in a reduced camera system has shown improvement in both time and memory requirements [33, 36]. Work has also been done by decoupling the original problem into several subproblems and applying BA in parallel [37].

The other way to make BA feasible in large-scale applications is to reduce the size of the BA itself. It can be done by reducing the number of frames by only selecting key-frames [38] or by reducing the number of variables to optimize [39]. Incremental Bundle Adjustment is another approach to reducing the size of the BA. Zhand et al. performed the reduced BA

by optimizing the image triplets locally and eliminating structure parameters [40]. Other works are done in the incremental BA by considering the camera and points within a specific time range. Engels et al. [41] applied the local BA when adding a new frame, whereas Mouragnon et al. [42] applied local BA when a new key-frame was added. Indelman et al. extend this idea to implement incremental light BA, where they only optimize the camera poses by algebraically removing the 3D points from the optimization and formulating the cost function in terms of three-view instead of reprojection errors [43].

This current work tends to mitigate that problem by using BA in a small window of texel images and optimizing the parameters within the window. This method is referred to as the Streaming Bundle Adjustment (SBA). Another difference between this work and the previous is the use of a bistatic Texel Camera instead of a coboresighted Texel Camera. The physical separation between the lidar sensor and the digital camera introduces parallax, which needs to be considered in the optimization. It also introduces two distinct normalized planes for lidar and image sensors.

Since this study works on a continuous input stream of texel images, the SBA algorithm is used to optimize and register texel images. The overall flowchart for this work is given in Fig 3.1. This work creates the output TDSM using these optimized texel images.

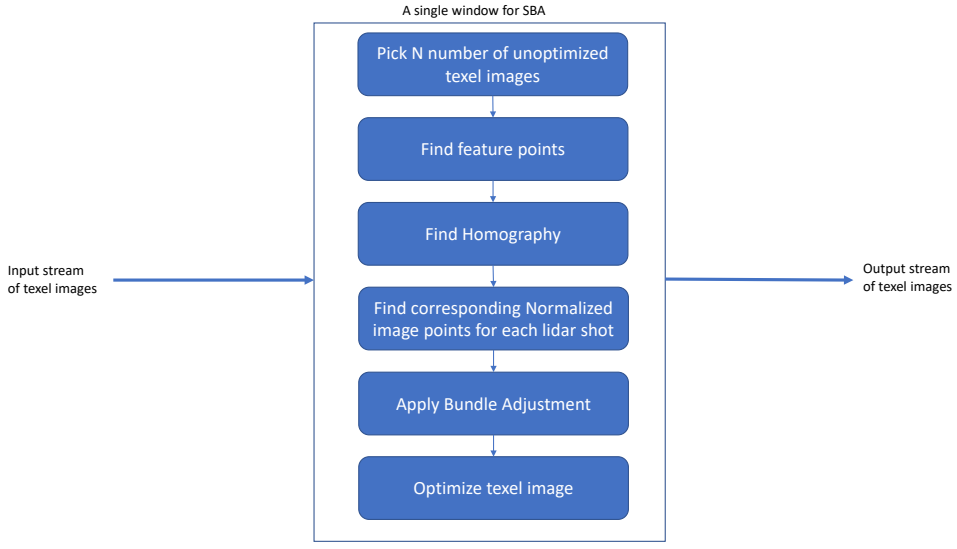


Fig. 3.1: Flowchart showing registration of texel images using SBA algorithm

3.2 Texel Registration

To create an accurate TDSM, the system must provide accurate point clouds, which can be obtained by exploiting the properties of a texel image. Registration of several texel images generates accurate point clouds of a vast area, which can be textured to generate the final TDSM [32]. For a texel image, each 3D point captured by that image has a corresponding projection on the 2D image using the mapping that was found during calibration [30]. If the adjacent texel image significantly overlaps with the current texel image, some of the 3D points can be projected on the 2D image of the adjacent texel image. The main steps in texel registration are:

1. Estimate pairwise homography between adjacent texel images.
2. Form the Projection matrix. This matrix defines the projection of each lidar measurement into the neighboring texel images.
3. Apply Bundle Adjustment to optimize 3D points and image pose.
4. Find appropriate texture and create TDSM.

In this work, we mainly focused on reducing the computation in the BA step, the memory storage usage in BA, and the size of the projection matrix. Each texel image j has its own 3D lidar points, a set denoted as I_j. Let $\mathbf{b}_i = [b_{ix}, b_{iy}, b_{iz}]^T$ be the i^{th} 3D lidar point in the world coordinate frame. Let $\mathbf{a}_j = [q_{j0}, q_{j1}, q_{j2}, q_{j3}, t_{jx}, t_{jy}, t_{jz}]^T$ be the camera pose for the texel image j, which consists of a quaternion derived from the rotation matrix and the translation vector. This pose defines the transformation from the j^{th} camera coordinate

frame to the world coordinate frame. We have,

(x_{ij}, y_{ij}) = true normalized image projections of b_i in the j^{th} texel image using calibration and image processing,

$(\hat{x}_{ij}, \hat{y}_{ij})$ = estimated normalized image projections of b_i in the j^{th} texel image using 3D points and pose,

λ_{ij} = true range of b_i captured by j^{th} texel image,

and $\hat{\lambda}_{ij}$ = estimated range of b_i using 3D points and pose in j^{th} texel image.

Let g_{ij} be the function that takes camera pose parameter $\mathbf{a}_j$ and 3D point $\mathbf{b}_i$ as the input and computes estimated image projections and the range. This function is highly non-linear and can be represented as the combination of the following operations.

1. **Transformation from world frame to the lidar coordinate frame**:

   Transformation of $\mathbf{b}_i$ from world frame to j^{th} lidar frame is performed using inverse pose composed using $\mathbf{a}_j$ as,

$$\begin{bmatrix} \hat{\chi}_{ij_x} \\ \hat{\chi}_{ij_y} \\ \hat{\chi}_{ij_z} \end{bmatrix} = \begin{bmatrix} 1 - \frac{2(q_{j2}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j2}-q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j3}+q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \\ \frac{2(q_{j1}q_{j2}+q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1 - \frac{2(q_{j1}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}-q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \\ \frac{2(q_{j1}q_{j3}-q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}+q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1 - \frac{2(q_{j2}^2+q_{j1}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \end{bmatrix}^T \begin{bmatrix} b_{ix} - t_{jx} \\ b_{iy} - t_{jy} \\ b_{iz} - t_{jz} \end{bmatrix}, \tag{3.1}$$

2. **Project to normalized lidar plane**:

   The point can then be projected onto the j^{th} normalized lidar plane by dividing by the z-coordinate, giving the 2D lidar projection $\tilde{x}_{ij}$ and $\tilde{y}_{ij}$. The range to the lidar

point $\hat{\lambda}_{ij}$ is the Euclidean distance between the camera center and the lidar point. These can be combined into a vector given by

$$\tilde{\boldsymbol{X}}_{ij} = \begin{bmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \\ \hat{\lambda}_{ij} \end{bmatrix} = \begin{bmatrix} \dfrac{\hat{\chi}_{ij_x}}{\hat{\chi}_{ij_z}} \\ \dfrac{\hat{\chi}_{ij_y}}{\hat{\chi}_{ij_z}} \\ \sqrt{\hat{\chi}_{ij_x}^2 + \hat{\chi}_{ij_y}^2 + \hat{\chi}_{ij_z}^2} \end{bmatrix}. \tag{3.2}$$

3. **Parallax correction**: Due to the physical separation between the two sensors, the parallax only occurs in the y-direction (in-track direction), which must be corrected before transforming into the normalized camera plane. The parallax correction, Δy, is given as

$$\Delta y = d_0 * \left(\frac{1.0}{z_{map}} + \frac{1.0}{\hat{\chi}_{ij_z}} \right), \tag{3.3}$$

where d_0 is the distance between two sensors and z_{map} is the distance from the texel camera to the calibration plane during mapping [27]. The updated $\tilde{y}_{ij}$ is given as

$$\tilde{y}_{ij} = \tilde{y}_{ij} - \Delta y. \tag{3.4}$$

4. **Transform normalized lidar projection to normalized image projection**: The normalized lidar projections are transformed into row-column using the calibration parameters in (3.5), which are then transformed to normalized image projection using camera matrix, $\mathbf{K}$. The row-column coordinate of the image is given by

$$
\begin{aligned}
dcol &= g_1 + g_2\tilde{x}_{ij} + g_3\tilde{x}_{ij}^2 + g_4\tilde{x}_{ij}^3 + g_5\tilde{y}_{ij} + g_6\tilde{x}_{ij}\tilde{y}_{ij} + g_7\tilde{y}_{ij}^2 + g_8\tilde{x}_{ij}\tilde{y}_{ij}^2 \\
&\quad + g_9\tilde{x}_{ij}^5 + g_{10}\tilde{x}_{ij}\tilde{y}_{ij}^4 + g_{11}\tilde{x}_{ij}^3\tilde{y}_{ij}^2, \\
drow &= h_1 + h_2\tilde{y}_{ij} + h_3\tilde{y}_{ij}^2 + h_4\tilde{y}_{ij}^3 + h_5\tilde{x}_{ij} + h_6\tilde{y}_{ij}\tilde{x}_{ij} + h_7\tilde{x}_{ij}^2 + h_8\tilde{y}_{ij}\tilde{x}_{ij}^2 \\
&\quad + h_9\tilde{y}_{ij}^5 + h_{10}\tilde{y}_{ij}\tilde{x}_{ij}^4 + h_{11}\tilde{y}_{ij}^3\tilde{x}_{ij}^2,
\end{aligned}
$$

and the normalized image projection is given by

$$
\begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \end{bmatrix} = \mathbf{K}^{-1} * \begin{bmatrix} dcol \\ drow \\ 1 \end{bmatrix}. \tag{3.5}
$$

And finally, the projection-range representation for 3D point $\mathbf{b}_i$ projected onto j^{th} texel image, $\hat{\boldsymbol{X}}_{ij}$, is given as

$$
\hat{\boldsymbol{X}}_{ij} = g_{ij}(\mathbf{a}_j, \mathbf{b}_i) = \begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \\ \hat{\lambda}_{ij} \end{bmatrix}. \tag{3.6}
$$

The projection error is computed by finding the Euclidean distance between the measured projection and the estimated projection in the camera projection plane, whereas the range error is computed using the difference between the measured range and the estimated range. The range error is computed only when the measured range is available. True range, λ_{ij}, is available only when 3D point, $\mathbf{b}_i$, is captured by texel image j. It should be noted that $\hat{x}_{ij}$ and $\hat{y}_{ij}$ are measurements in the camera frame. In contrast, $\hat{\lambda}_{ij}$ is a measurement in the lidar frame. The error in the system, ϵ_{ij}, is dependent upon the relationship between point i and texel image j and has, therefore, two cases:

1. If $\mathbf{b_i} \in \mathbf{I_j}$ (owner of $\mathbf{b}_i$ is texel image j)

   In this case, the true range is measured by the texel camera, and normalized image projection is calculated from the calibration. If $[x_{ij}, y_{ij}, \lambda_{ij}]$ are the normalized image projections and the true range, then the square of the error, ϵ_{ij}^2, can be written as

$$\epsilon_{ij}^2 = \frac{1}{\sigma_I^2}(x_{ij} - \hat{x}_{ij})^2 + \frac{1}{\sigma_I^2}(y_{ij} - \hat{y}_{ij})^2 + \frac{1}{\sigma_\lambda^2}(\lambda_{ij} - \hat{\lambda}_{ij})^2, \tag{3.7}$$

   where σ_I^2, and σ_λ^2 are the error variances on the 2D calibrated projected points and the range measurements respectively.

2. If $\mathbf{b_i} \notin \mathbf{I_j}$ (owner of $\mathbf{b}_i$ is not texel image j)

   In this case, the true projection is found using image processing. Since there is no information on true range measurement, it is excluded from the error. The error, ϵ_{ij}^2, in this case, is given as

$$\epsilon_{ij}^2 = \frac{1}{\sigma_{\bar{I}}^2}(x_{ij} - \hat{x}_{ij})^2 + \frac{1}{\sigma_{\bar{I}}^2}(y_{ij} - \hat{y}_{ij})^2, \tag{3.8}$$

   where $\sigma_{\bar{I}}^2$ is the error variances on the 2D projection points found using image processing [32].

3.3 Conventional Bundle Adjustment

The low-cost INS unit available on the texel camera provides only the coarse attitude and position information far below the required accuracy. The final optimization is needed to correct the data obtained from the texel image. The correction is done on the position of 3D points and the location of the camera in the world coordinate frame. A non-linear optimization can be performed to jointly refine the position of lidar points and the camera attitude using the projection of lidar points onto neighboring texel images and the range data. Bundle adjustment tries to minimize the error cost function, which depends on the

observed true projections and ranges and the predicted projections and ranges re-projected from the 3D structure using camera poses. Conventional Bundle Adjustment (CBA) is done as the last step taking account of all the cameras and 3D points and optimizing jointly.

Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be the function that takes all camera parameters and 3D points as the input and computes image projections on each image and the ranges. Specifically, given the initial estimate, $P_0 \in \mathbb{R}^m$, and measurement vector $X \in \mathbb{R}^n$, we seek to find the $\hat{P}$ that satisfies $X = f(\hat{P}) - \epsilon$ such that $||\epsilon||^2$ is minimized. For CBA, the squared norm of the error, $||\epsilon||^2$, is computed by combing (3.7) and (3.8) and is given as

$$||\epsilon||^2 = \sum_{j=1}^{M} \left(\sum_{\substack{i=1 \\ i \in I_j}}^{N} \frac{1}{\sigma_I^2}[(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2] + \sum_{\substack{i=1 \\ i \notin I_j}}^{N} \frac{1}{\sigma_{\bar{I}}^2}[(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2] + \sum_{\substack{i=1 \\ i \in I_j}}^{N} \frac{1}{\sigma_\lambda^2}(\lambda_{ij} - \hat{\lambda}_{ij})^2 \right), \tag{3.9}$$

where **M** is the total number of texel images and **N** is the total number of 3D points that are being optimized. If any of the measurements are missing, that term is simply omitted from the sum. For example, if the projection of a 3D point, $\mathbf{b}_k$, is not seen by the l^{th} camera, then the term containing (x_{kl}, y_{kl}) are omitted from the total sum. In CBA, the parameter vector consists of all the image poses for each texel image, $\mathbf{a}_j$, and 3D points captured by all texel images, $\mathbf{b}_i$. The measurement vector consists of all the image projections of $\mathbf{b}_i$ on each image, either through calibration or image processing, and the measured range for each $\mathbf{b}_i$. The total number of parameters that are optimized is $(7M + 3N)$.

The optimization goal is to reduce the error given in (3.9). One of the techniques to perform BA is to use the Levenberg-Marquardt Algorithm (LMA). LMA is an iterative algorithm that adjusts the camera parameter $\mathbf{a}_j$ and 3D points $\mathbf{b}_i$ as the algorithm proceeds. The initial poses obtained from the INS are used as the seed of optimization to converge faster.

LMA is a hybrid between Gauss-Newton and Gradient descent algorithms and varies between them with the help of a damping parameter. It is an iterative algorithm where the parameters are updated in each iteration to reduce the error in the system. if $\hat{\mathbf{X}} = f(\hat{\mathbf{P}})$ and ϵ is the error between $\mathbf{X}$ and $\hat{\mathbf{X}}$, then the central normal equation for LMA is given as,

$$(\mathbf{J}^T\boldsymbol{\Sigma}^{-1}\mathbf{J} + \rho\,\mathrm{diag}(\mathbf{J}^T\boldsymbol{\Sigma}^{-1}\mathbf{J}))\delta\mathbf{x} = \mathbf{J}^T\boldsymbol{\Sigma}^{-1}\epsilon, \tag{3.10}$$

where $\mathbf{J} = \partial f/\partial \mathbf{P}$ is the Jacobian, $\boldsymbol{\Sigma}$ is the covariance matrix of the measurement, $\delta\mathbf{x}$ is the parameter update vector, and ρ is the damping parameter. The parameter vector is partitioned between camera parameters and 3D points to take advantage of the sparsity in the Jacobian [7]. If the parameter is partitioned as $\mathbf{P} = [\mathbf{a}^T, \mathbf{b}^T]^T$, then the Jacobian has the form $J = [A|B]$ where

$$A = \frac{\partial \hat{X}}{\partial \mathbf{a}}$$
$$B = \frac{\partial \hat{X}}{\partial \mathbf{b}}.$$

If the update vector $\delta\mathbf{x}$ is also partitioned as $\delta\mathbf{x} = [\delta\mathbf{a}|\delta\mathbf{b}]$, then the un-augmented normal equation is given as

$$\begin{bmatrix} A^T\boldsymbol{\Sigma}^{-1}A & A^T\boldsymbol{\Sigma}^{-1}B \\ B^T\boldsymbol{\Sigma}^{-1}A & B^T\boldsymbol{\Sigma}^{-1}B \end{bmatrix} \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} = \begin{pmatrix} A^T\boldsymbol{\Sigma}^{-1}\epsilon \\ B^T\boldsymbol{\Sigma}^{-1}\epsilon \end{pmatrix}$$
$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix} \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{A}} \\ \epsilon_{\mathbf{B}} \end{pmatrix}. \tag{3.11}$$

The factor $(1 + \rho)$ is augmented to the diagonal element of the LHS matrix of (3.11) to give

$$\begin{bmatrix} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{bmatrix} \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{A}} \\ \epsilon_{\mathbf{B}} \end{pmatrix}. \tag{3.12}$$

The left-hand side of the (3.12) has a sparse structure. This is because of the fact that the measurement, $\hat{\boldsymbol{X}}_{ij}$, only depends on $\mathbf{a}_j$ and $\mathbf{b}_i$. This means, $\partial\hat{\boldsymbol{X}}_{ij}/\partial\mathbf{a}_k = 0$ unless $j = k$. Similarly, $\partial\hat{\boldsymbol{X}}_{ij}/\partial\mathbf{b}_k = 0$ unless $i = k$. Equation (3.12) is solved with a sparse matrix solver to get the value of $[\delta\mathbf{a}, \delta\mathbf{b}]$ which is then applied to the parameter as $P = [a + \delta\mathbf{a}, b + \delta\mathbf{b}]$. Suppose the value of the total error given in (3.9) is reduced by using the new parameter value. In that case, the increment is accepted, and ρ is divided by a factor of 10 before the next iteration. On the other hand, if the error is increased after using the updated parameter value, then ρ is multiplied by a factor of 10, and the normal equation given in (3.12) is solved again until the update vector, $[\delta\mathbf{a}, \delta\mathbf{b}]$, is found when the reduction of the error occurred.

3.4 Streaming Bundle Adjustment

In the SBA algorithm, instead of applying bundle adjustment to all the texel images, a sliding window is used and a local bundle adjustment is applied only to the texel images inside the window. Once the parameters for that window are optimized, old texel images are removed from the window and new images are added into the window.

The size of the window is chosen based on the overlapping of one texel image onto the other. In this technique, texel images are added to the existing 3D structure as the algorithm proceeds. A variable named *look length* and denoted as $\mathcal{L}$, defines the effect of one texel image onto the other, where $\mathcal{L}$ is the number of texel images after which the current image has no effect. More precisely, changing the parameters of texel image I_j doesn't change the parameters of texel images after $I_{j+\mathcal{L}}$ and before $I_{j-\mathcal{L}}$. To reduce the computation further, $\mathcal{L}$ texel images are optimized in each window instead of one image. The window length is set as $3\mathcal{L}$. In each iteration of SBA, $\mathcal{L}$ texel images in the middle of

the window are optimized. For each window, initial $\mathcal{L}$ texel images are already optimized in the previous iteration and are called *past* images. The middle $\mathcal{L}$ texel images are optimized in the current iteration and are called *present* images. The final $\mathcal{L}$ texel images are called $future$ texel images as shown in Fig 3.2.

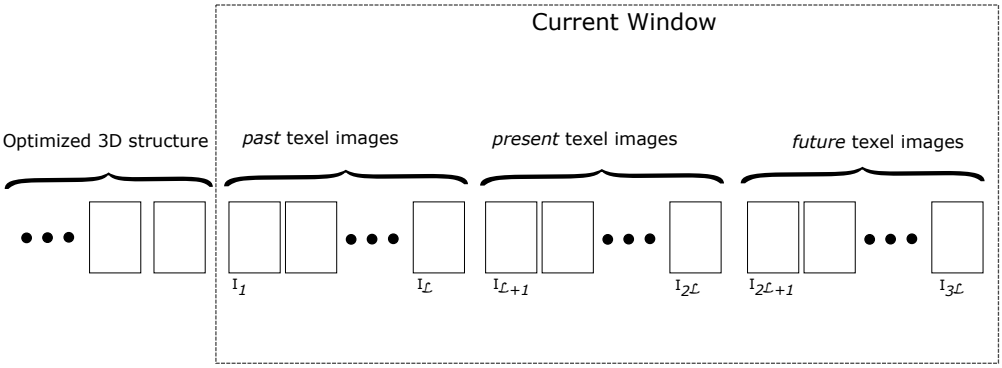


Fig. 3.2: SBA operation

In the current window of the SBA, the parameter vector consists of all the image poses $\mathbf{a}_j$ and 3D points $\mathbf{b}_i$ from the *present* and the $future$ texel images. Let m be the total number of *present* and $future$ texel images in the current window such that $m = 2\mathcal{L}$. If n is the total 3D points captured by *present* and $future$ texel images, the total parameters optimized in one window iteration are $(7m+3n)$. Note that this number is much smaller than the total parameters optimized in the CBA. It also keeps the computation load approximately constant as more texel images are acquired.

In each iteration of SBA, the measurement vector consists of only the measurement from the current window. It includes the image projections of all 3D points captured by *present* and $future$ texel images onto all texel images. It also includes the measured range for the *present* and the $future$ texel images. Finally, it includes the image projections of 3D points captured by *past* texel images onto *present* texel images.

At the end of the current iteration of the SBA, *present* texel images are optimized and the *past* texel images are added to the 3D structure. Even though *present* texel images are

optimized in each iteration, the parameters for the $future$ texel images are also changed. This makes the next iteration of the SBA converge faster, as the $future$ images will have already been partially optimized in the current iteration. For the next iteration, *present* texel images become *past*, $future$ texel images become *present* and $\mathcal{L}$ new texel images are added to the window as $future$ images and the process continues until all texel images are optimized.

The cost function that the SBA tries to minimize for each window is given as follows:

$$
\begin{aligned}
||\epsilon||^2 = &\sum_{j=1}^{3\mathcal{L}} \sum_{\substack{i \\ b_i \in I_{present} \\ b_i \in I_{future} \\ b_i \in I_j}} \left[\frac{1}{\sigma_I^2} [(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2] + \frac{1}{\sigma_\lambda^2} (\lambda_{ij} - \hat{\lambda}_{ij})^2 \right] \\
&+ \sum_{j=1}^{3\mathcal{L}} \sum_{\substack{i \\ b_i \in I_{present} \\ b_i \in I_{future} \\ b_i \notin I_j}} \frac{1}{\sigma_I^2} [(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2] \\
&+ \sum_{j=\mathcal{L}+1}^{2\mathcal{L}} \sum_{\substack{i \\ b_i \in I_{past}}} \frac{1}{\sigma_I^2} [(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2].
\end{aligned}
\tag{3.13}
$$

The optimization routine to do the SBA is the same as the CBA, except the matrix that needs to be inverted is much smaller in the SBA compared to CBA. The inversion of a small matrix makes the SBA algorithm much faster than the conventional one. At the start of the algorithm, the CBA is employed to start constructing the 3D structure. Once the 3D structure is built with the optimized texel images, SBA is employed, and more texel images are optimized and added to the structure to create the final TDSM.

3.5 Illustration example

Let us consider an example where there are five texel images I_j, each with a pose $\mathbf{a}_j$, and six 3D points, $\mathbf{b}_i$, such that, $\mathbf{b}_1$ is captured by I_1, $\mathbf{b}_2$ is captured by I_2, $\mathbf{b}_3$ and $\mathbf{b}_4$ are captured by I_3, $\mathbf{b}_5$ is captured by I_4, and finally, $\mathbf{b}_6$ is captured by I_5. We can write,

$$\mathbf{b}_1 \in \mathbf{a}_1,\ \mathbf{b}_2 \in \mathbf{a}_2,\ \mathbf{b}_3, \mathbf{b}_4 \in \mathbf{a}_3,\ \mathbf{b}_5 \in \mathbf{a}_4,\ \mathbf{b}_6 \in \mathbf{a}_5$$

- For CBA

  Let $\boldsymbol{X_{ij}}$ be the true projection-range coordinate for i^{th} 3D points onto j^{th} texel image. Note that the true range is only available if i^{th} 3D point is captured by j^{th} texel image. The projection matrix stores the true image projection and range value for all 3D points. In the actual implementation of CBA, this matrix is large and uses a huge amount of disk storage.

Table 3.1: Projection Matrix for CBA

	a_1	a_2	a_3	a_4	a_5
b_1	$\boldsymbol{X_{11}}$ $[x_{11}, y_{11}, \lambda_{11}]$	$\boldsymbol{X_{12}}$ $[x_{12}, y_{12}, 0]$	$\boldsymbol{X_{13}}$ $[x_{13}, y_{13}, 0]$	$\boldsymbol{X_{14}}$ $[x_{14}, y_{14}, 0]$	$\boldsymbol{X_{15}}$ $[x_{15}, y_{15}, 0]$
b_2	$\boldsymbol{X_{21}}$ $[x_{21}, y_{21}, 0]$	$\boldsymbol{X_{22}}$ $[x_{22}, y_{22}, \lambda_{22}]$	$\boldsymbol{X_{23}}$ $[x_{23}, y_{23}, 0]$	$\boldsymbol{X_{24}}$ $[x_{24}, y_{24}, 0]$	$\boldsymbol{X_{25}}$ $[x_{25}, y_{25}, 0]$
b_3	$\boldsymbol{X_{31}}$ $[x_{31}, y_{31}, 0]$	$\boldsymbol{X_{32}}$ $[x_{32}, y_{32}, 0]$	$\boldsymbol{X_{33}}$ $[x_{33}, y_{33}, \lambda_{33}]$	$\boldsymbol{X_{34}}$ $[x_{34}, y_{34}, 0]$	$\boldsymbol{X_{35}}$ $[x_{35}, y_{35}, 0]$
b_4	$\boldsymbol{X_{41}}$ $[x_{41}, y_{41}, 0]$	$\boldsymbol{X_{42}}$ $[x_{42}, y_{42}, 0]$	$\boldsymbol{X_{43}}$ $[x_{43}, y_{43}, \lambda_{43}]$	$\boldsymbol{X_{44}}$ $[x_{44}, y_{44}, 0]$	$\boldsymbol{X_{45}}$ $[x_{45}, y_{45}, 0]$
b_5	$\boldsymbol{X_{51}}$ $[x_{51}, y_{51}, 0]$	$\boldsymbol{X_{52}}$ $[x_{52}, y_{52}, 0]$	$\boldsymbol{X_{53}}$ $[x_{53}, y_{53}, 0]$	$\boldsymbol{X_{54}}$ $[x_{54}, y_{54}, \lambda_{54}]$	$\boldsymbol{X_{55}}$ $[x_{55}, y_{55}, 0]$
b_6	$\boldsymbol{X_{61}}$ $[x_{61}, y_{61}, 0]$	$\boldsymbol{X_{62}}$ $[x_{62}, y_{62}, 0]$	$\boldsymbol{X_{63}}$ $[x_{63}, y_{63}, 0]$	$\boldsymbol{X_{64}}$ $[x_{64}, y_{64}, 0]$	$\boldsymbol{X_{65}}$ $[x_{65}, y_{65}, \lambda_{65}]$

CBA tries to minimize the cost function as

$$||\epsilon||^2 = \sum_{j=1}^{5} \left(\sum_{\substack{i=0 \\ b_i \in I_j}}^{6} \frac{1}{\sigma_I^2}[(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2] + \sum_{\substack{i=0 \\ b_i \notin I_j}}^{6} \frac{1}{\sigma_{\bar{I}}^2}[(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2] + \sum_{\substack{i=0 \\ b_i \in I_j}}^{6} \frac{1}{\sigma_\lambda^2}(\lambda_{ij} - \hat{\lambda}_{ij})^2 \right). \tag{3.14}$$

Table 3.2: Jacobian matrix for CBA

$J = \frac{\partial \hat{X}}{\partial P} =$

	a_1	a_2	a_3	a_4	a_5	b_1	b_2	b_3	b_4	b_5	b_6
$\boldsymbol{X_{11}}$	A_{11}	0	0	0	0	B_{11}	0	0	0	0	0
$\boldsymbol{X_{12}}$	0	A_{12}	0	0	0	B_{12}	0	0	0	0	0
$\boldsymbol{X_{13}}$	0	0	A_{13}	0	0	B_{13}	0	0	0	0	0
$\boldsymbol{X_{14}}$	0	0	0	A_{14}	0	B_{14}	0	0	0	0	0
$\boldsymbol{X_{15}}$	0	0	0	0	A_{15}	B_{15}	0	0	0	0	0
$\boldsymbol{X_{21}}$	A_{21}	0	0	0	0	0	B_{21}	0	0	0	0
$\boldsymbol{X_{22}}$	0	A_{22}	0	0	0	0	B_{22}	0	0	0	0
$\boldsymbol{X_{34}}$	0	0	A_{23}	0	0	0	B_{23}	0	0	0	0
$\boldsymbol{X_{24}}$	0	0	0	A_{24}	0	0	B_{24}	0	0	0	0
$\boldsymbol{X_{25}}$	0	0	0	0	A_{25}	0	B_{25}	0	0	0	0
$\boldsymbol{X_{31}}$	A_{31}	0	0	0	0	0	0	B_{31}	0	0	0
$\boldsymbol{X_{32}}$	0	A_{32}	0	0	0	0	0	B_{32}	0	0	0
$\boldsymbol{X_{33}}$	0	0	A_{33}	0	0	0	0	B_{33}	0	0	0
$\boldsymbol{X_{34}}$	0	0	0	A_{34}	0	0	0	B_{34}	0	0	0
$\boldsymbol{X_{35}}$	0	0	0	0	A_{35}	0	0	B_{35}	0	0	0
$\boldsymbol{X_{41}}$	A_{41}	0	0	0	0	0	0	0	B_{41}	0	0
$\boldsymbol{X_{42}}$	0	A_{42}	0	0	0	0	0	0	B_{42}	0	0
$\boldsymbol{X_{43}}$	0	0	A_{43}	0	0	0	0	0	B_{43}	0	0
$\boldsymbol{X_{44}}$	0	0	0	A_{44}	0	0	0	0	B_{44}	0	0
$\boldsymbol{X_{45}}$	0	0	0	0	A_{45}	0	0	0	B_{45}	0	0
$\boldsymbol{X_{51}}$	A_{51}	0	0	0	0	0	0	0	0	B_{51}	0
$\boldsymbol{X_{52}}$	0	A_{52}	0	0	0	0	0	0	0	B_{52}	0
$\boldsymbol{X_{53}}$	0	0	A_{53}	0	0	0	0	0	0	B_{53}	0
$\boldsymbol{X_{54}}$	0	0	0	A_{54}	0	0	0	0	0	B_{54}	0
$\boldsymbol{X_{55}}$	0	0	0	0	A_{55}	0	0	0	0	B_{55}	0
$\boldsymbol{X_{61}}$	A_{61}	0	0	0	0	0	0	0	0	0	B_{61}
$\boldsymbol{X_{62}}$	0	A_{62}	0	0	0	0	0	0	0	0	B_{62}
$\boldsymbol{X_{63}}$	0	0	A_{63}	0	0	0	0	0	0	0	B_{63}
$\boldsymbol{X_{64}}$	0	0	0	A_{64}	0	0	0	0	0	0	B_{64}
$\boldsymbol{X_{65}}$	0	0	0	0	A_{65}	0	0	0	0	0	B_{65}

The Jacobian matrix in the LMA algorithm for CBA is given in Table 3.2. A_{ij} and B_{ij} in this table are the Jacobians such that

$$A_{ij} = \frac{\partial \hat{X_{ij}}}{\partial \mathbf{a}_i}$$

and

$$B_{ij} = \frac{\partial \hat{X_{ij}}}{\partial \mathbf{b}_j}.$$

As discussed before, the sparse nature of the Jacobian can be seen as most of the entries in the table are zero.

- SBA

If we consider $\mathcal{L} = 1$, then the length of the window is $3\mathcal{L} = 3 * 1 = 3$. Since the optimized 3D structure is necessary for SBA, the first iteration is completed using CBA on just three images.

Iteration 1

$\mathbf{b}_1 \in \mathbf{a}_1,\ \mathbf{b}_2 \in \mathbf{a}_2,\ \mathbf{b}_3, \mathbf{b}_4 \in \mathbf{a}_3$

The projection matrix is shown in Table 3.3

Table 3.3: Projection matrix for SBA, Iteration 1

	a_1	a_2	a_3		
b_1	$\boldsymbol{X_{11}}$ $[x_{11}, y_{11}, \lambda_{11}]$	$\boldsymbol{X_{12}}$ $[x_{12}, y_{12}, 0]$	$\boldsymbol{X_{13}}$ $[x_{13}, y_{13}, 0]$		
b_2	$\boldsymbol{X_{21}}$ $[x_{21}, y_{21}, 0]$	$\boldsymbol{X_{22}}$ $[x_{22}, y_{22}, \lambda_{22}]$	$\boldsymbol{X_{23}}$ $[x_{23}, y_{23}, 0]$	$\Rightarrow$	Optimize $\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2$
b_3	$\boldsymbol{X_{31}}$ $[x_{31}, y_{31}, 0]$	$\boldsymbol{X_{32}}$ $[x_{32}, y_{32}, 0]$	$\boldsymbol{X_{33}}$ $[x_{33}, y_{33}, \lambda_{33}]$		
b_4	$\boldsymbol{X_{41}}$ $[x_{41}, y_{41}, 0]$	$\boldsymbol{X_{42}}$ $[x_{42}, y_{42}, 0]$	$\boldsymbol{X_{43}}$ $[x_{43}, y_{43}, \lambda_{43}]$		

In the next iteration, we slide the window to include texel image I_4, and image pose $\mathbf{a}_1$ corresponding to texel image I_1 is considered optimized and is added to the 3D structure.

Iteration 2

$\mathbf{b}_2 \in \mathbf{a}_2,\ \mathbf{b}_3, \mathbf{b}_4 \in \mathbf{a}_3\ \mathbf{b}_5 \in \mathbf{a}_4$

As seen in Table 3.4, the size of the matrix remains the same. Most of the measurements for the true projection and the range are either already computed in the previous iteration. Since the measurement vector doesn't include the projection of 3D points from *past* texel images onto *past* or *future* texel images and the projection of 3D points from *future* texel images onto *past* texel images, elements corresponding to those are not used in the optimization. This saves significant computation from recalculating the true projection and saving in memory, as the projection matrix for each iteration of SBA is much smaller than CBA.

Table 3.4: Projection matrix for SBA, Iteration 2

	a_2	a_3	a_4		
b_2	$\boldsymbol{X_{22}}$ (not used)	$\boldsymbol{X_{23}}$ (pre-computed)	$\boldsymbol{X_{24}}$ (not used)		
b_3	$\boldsymbol{X_{32}}$ (pre-computed)	$\boldsymbol{X_{33}}$ (pre-computed)	$\boldsymbol{X_{34}}$ $[x_{34}, y_{34}, 0]$	$\Rightarrow$	Optimize $\mathbf{a}_3, \mathbf{b}_3, \mathbf{b}_4$
b_4	$\boldsymbol{X_{42}}$ (pre-computed)	$\boldsymbol{X_{43}}$ (pre-computed)	$\boldsymbol{X_{44}}$ $[x_{44}, y_{44}, 0]$		
b_5	$\boldsymbol{X_{52}}$ (not used)	$\boldsymbol{X_{53}}$ $[x_{53}, y_{53}, 0]$	$\boldsymbol{X_{54}}$ $[x_{54}, y_{54}, \lambda_{54}]$		

Iteration 3

$\mathbf{b}_3, \mathbf{b}_4 \in \mathbf{a}_3,\ \mathbf{b}_5 \in \mathbf{a}_4,\ \mathbf{b}_6 \in \mathbf{a}_5$

As we can see in Table 3.5, again the large portion of the projection matrix is re-used for the next iteration, thus saving computation.

The Jacobian matrix for SBA is similar to CBA, except the rows corresponding to the measurements not considered in the optimization are missing. Also, the columns that correspond to the already optimized parameters are missing from the Jacobian matrix. The Jacobian matrix for Iteration 1 is the same as the CBA case, except that the size is much smaller as we only consider $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$. The Jacobian for iterations 2

Table 3.5: Projection matrix for SBA, Iteration 3

	a_3	a_4	a_5
b_3	$\boldsymbol{X_{33}}$ (not used)	$\boldsymbol{X_{34}}$ (pre-computed)	$\boldsymbol{X_{35}}$ (not used)
b_4	$\boldsymbol{X_{43}}$ (not used)	$\boldsymbol{X_{44}}$ (pre-computed)	$\boldsymbol{X_{45}}$ (not used)
b_5	$\boldsymbol{X_{53}}$ (pre-computed)	$\boldsymbol{X_{54}}$ (pre-computed)	$\boldsymbol{X_{55}}$ $[x_{55}, y_{55}, 0]$
b_6	$\boldsymbol{X_{63}}$ (not used)	$\boldsymbol{X_{64}}$ $[x_{53}, y_{64}, 0]$	$\boldsymbol{X_{65}}$ $[x_{65}, y_{65}, \lambda_{65}]$

$\Rightarrow$ Optimize $\mathbf{a}_4, \mathbf{b}_5$

and 3 for SBA is given in Table 3.6 and Table 3.7, respectively.

Table 3.6: Jacobian matrix for SBA, Iteration 2

$J = \frac{\partial \hat{X}}{\partial P} =$

	a_3	a_4	b_3	b_4	b_5
$\boldsymbol{X_{23}}$	A_{23}	0	0	0	0
$\boldsymbol{X_{32}}$	0	0	B_{32}	0	0
$\boldsymbol{X_{33}}$	A_{33}	0	B_{33}	0	0
$\boldsymbol{X_{34}}$	0	A_{34}	B_{34}	0	0
$\boldsymbol{X_{42}}$	0	0	0	B_{42}	0
$\boldsymbol{X_{43}}$	A_{43}	0	0	B_{43}	0
$\boldsymbol{X_{44}}$	0	A_{44}	0	B_{44}	0
$\boldsymbol{X_{53}}$	A_{53}	0	0	0	B_{53}
$\boldsymbol{X_{54}}$	0	A_{54}	0	0	B_{54}

3.6 Results and Discussion

The Center of Advanced Imaging Ladar (CAIL) has developed a camera called the texel camera, which is used to collect test data. The texel camera comprises a lidar sensor, a digital camera, an INS, and an onboard processor. The lidar sensor has 16 channels and can rotate 360 degrees at a rate of 10 Hz. The lidar has a vertical field of view of 15.8 degrees and a configurable horizontal field of view, usually set between 30 and 40 degrees. The camera's horizontal and vertical field of view are determined by the lidar parameters, and its maximum resolution is 2832 pixels in the vertical direction and 4240 pixels in the horizontal direction. The texel camera, depicted in Fig 3.3(a), can be mounted on any small

Table 3.7: Jacobian matrix for SBA, Iteration 3

$$J = \frac{\partial \hat{X}}{\partial P} =$$

	a_4	a_5	b_5	b_6
$\boldsymbol{X_{34}}$	A_{34}	0	0	0
$\boldsymbol{X_{44}}$	A_{44}	0	0	0
$\boldsymbol{X_{53}}$	0	0	B_{53}	0
$\boldsymbol{X_{54}}$	A_{54}	0	B_{54}	0
$\boldsymbol{X_{55}}$	0	A_{55}	B_{55}	0
$\boldsymbol{X_{64}}$	A_{64}	0	0	B_{64}
$\boldsymbol{X_{65}}$	0	A_{65}	0	B_{65}

unmanned aerial vehicle (SUAV) capable of carrying a payload of less than 2 kg. The texel camera that is used to capture data for this study is mounted onto the DJI rotorcraft as shown in Fig 3.3(b).

(a) (b)

Fig. 3.3: (a) Texel camera (b) Texel camera mounted in DJI rotorcraft for data acquisition

This aircraft was flown over Cutler Dam and the surrounding area in Cache Valley, Utah, for approximately 6 minutes. The drone was piloted to fly approximately 70 m above the terrain during the bulk of the flight. Each minute of flight time generates about 600 texel images. The inexpensive IMU sensor in the texel camera provides the GPS position and attitude (pose) of the texel camera during each acquisition. Because of its low cost, the quality of the pose is fairly coarse, with position to within 3 m and attitude to within

about 0.3 degrees in roll, pitch, and yaw.

Fig 3.4 shows the registration result for the Cutler data for 84 images. Fig 3.4(a) shows the result when no optimization is performed. It can be clearly seen that optimization is needed to create accurate TDSM. Fig 3.4(b) shows when CBA is employed for 84 images. Fig 3.4(c) shows the same images registered using SBA. For this result, $\mathcal{L}$ is set to be 7. For the first 21 images, CBA is employed to generate the optimized structure, and in each iteration of the SBA algorithm, seven texel images are added to the structure. The TDSM is generated at the end using all texel images optimized using SBA. The result for TDSM created using CBA and SBA looks visually similar.

The algorithm was run again for a different dataset. This was taken near Franklin, Utah, and 84 images were run again with the same parameter and the same $\mathcal{L}$. Fig 3.5 shows the result for Franklin data. Again the unregistered TDSM in Fig 3.5(a) is distorted and is not textured smoothly. It can be seen from Fig 3.5(b) and Fig 3.5(c) that optimization using both methods significantly improve the quality of the final TDSM.

The experiment was run again on Cutler data with a different $\mathcal{L}$. TDSM is created with a $\mathcal{L}$ of 2,3,4,6,7,8,10,12,21 and 28. The result for this is shown in Fig 3.6. Even though visually they look similar, a statistical analysis is performed to know the relationship between $\mathcal{L}$ and the quality of the final TDSM. To measure the quality of the registration result using SBA, the comparison is made with the TDSM obtained using CBA. Because of the absence of the surveyed points, the information about the true ground truth is not present. One method is to measure the distance between any two points in the TDSM generated using CBA and compare it with the same distance in the TDSM generated using SBA. This comparison can be made between several points.

Two thousand points are randomly selected from the registered TDSM using CBA, and pairwise Euclidean distances are measured. The total distance measure is $\binom{2000}{2} = 1.999 \times 10^6$ and is considered as the true distance. The same 2000 points are chosen in the unregistered TDSM as well the TDSM created using SBA with different $\mathcal{L}$, and the same 1.999×10^6 distances are found. For each pair of points, the error between the true distance

as measured from TDSM using CBA and the distance measured from TDSM using SBA is calculated. The mean error is computed by finding the mean of these errors. Similarly, the standard deviation and the mean square error (MSE) are also computed using these errors and are given in Table 3.8. The table shows that the MSE is low when the $\mathcal{L}$ is set at 7. By looking at the input texel images, it was seen that there was no significant overlap between texel images after 7^{th} adjacent image and no overlap after 10^{th} adjacent image. Making the window size larger when there is no overlap between images has a negative effect on the quality of TDSM. Theoretically, $\mathcal{L}$ can be computed using the FOV of the camera, the speed and altitude of the aircraft, and the rate of image acquisition.

TDSM created from texel images using CBA requires forming a projection table for all the images. This requires pairwise homography between two adjacent texel images, image correlation, and a good distribution of lidar shots in the image. The CBA method failed for texel images with fewer feature points and lidar shots. Fig 3.7 shows an example of such a condition. Fig 3.7(a) shows that at the end of the flight path, the aircraft reached the Cutler reservoir area, where there are fewer features and no returns from the lidar shot, as evident from the wireframe diagram. The TDSM using CBA failed in this case, whereas TDSM using SBA can save the optimized TDSM and can start creating a new TDSM once the features and lidar shots are available, as shown in Fig 3.7(b).

As discussed before, the main advantage of using SBA over CBA to create TDSM is the execution speed and the memory consumption. The experiment was done on a desktop computer with an AMD Ryzen 7 1700 Eight-Core Processor running at 3.00 GHz and 24 GB of DDR4 RAM. The computer was running Windows 10 Enterprise. This computer was also equipped with an NVIDIA GeForce GTX 1050 GPU. The processing was done in a C++ application, and multi-threading and CUDA were used to speed up the program when applicable. Table 3.9 shows the comparison between TDSM created using CBA and SBA in terms of time to complete and the maximum memory consumed during the whole algorithm. The overall algorithm includes finding projection points, forming a projection table, bundle adjustment, finding texture, and finally, creating the TDSM. It is evident that

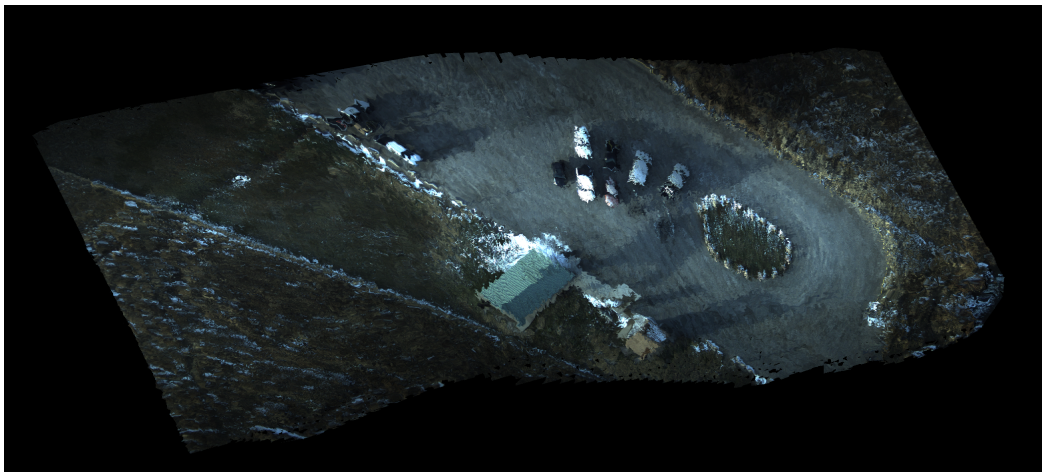
Table 3.8: Error comparison between non-registered TDSM and TDSM created using SBA with different $\mathcal{L}$

	Mean Error (m)	**Std Dev** (m)	**MSE** (m^2)
Unregistered TDSM	1.05823	1.27998	2.75819
SBA, $\mathcal{L} = 2$	0.04128	0.06071	0.00538
SBA, $\mathcal{L} = 3$	0.02836	0.05404	0.00372
SBA, $\mathcal{L} = 4$	0.01723	0.04626	0.00243
SBA, $\mathcal{L} = 6$	0.00626	0.03394	0.00119
SBA, $\mathcal{L} = 7$	0.00614	0.03292	0.00112
SBA, $\mathcal{L} = 8$	0.02147	0.04552	0.00253
SBA, $\mathcal{L} = 10$	0.02501	0.05773	0.00395
SBA, $\mathcal{L} = 12$	0.03226	0.05936	0.00456
SBA, $\mathcal{L} = 21$	0.04345	0.06211	0.00574
SBA, $\mathcal{L} = 28$	0.03432	0.05892	0.00464

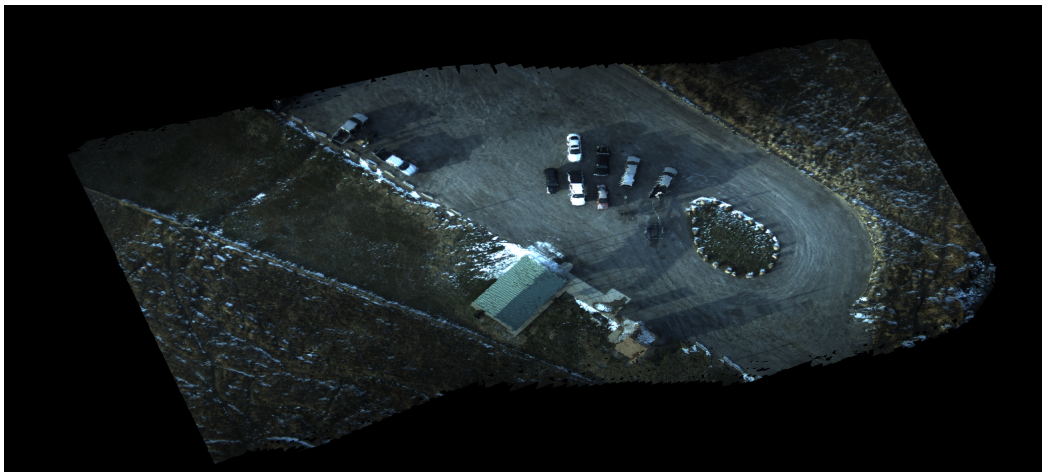
the memory consumption in SBA remains constant even after adding more texel images, whereas the memory consumption in CBA increases as more input texel images are added, and the algorithm cannot process after 150 texel images. The rate of increase for time to create TDSM using SBA is much slower than in the CBA case. The time to create TDSM using 150 texel images using CBA is almost the same as the time to create TDSM using 1000 texel images using SBA.

Table 3.9: Memory consumption comparison between SBA and CBA for different *look length*

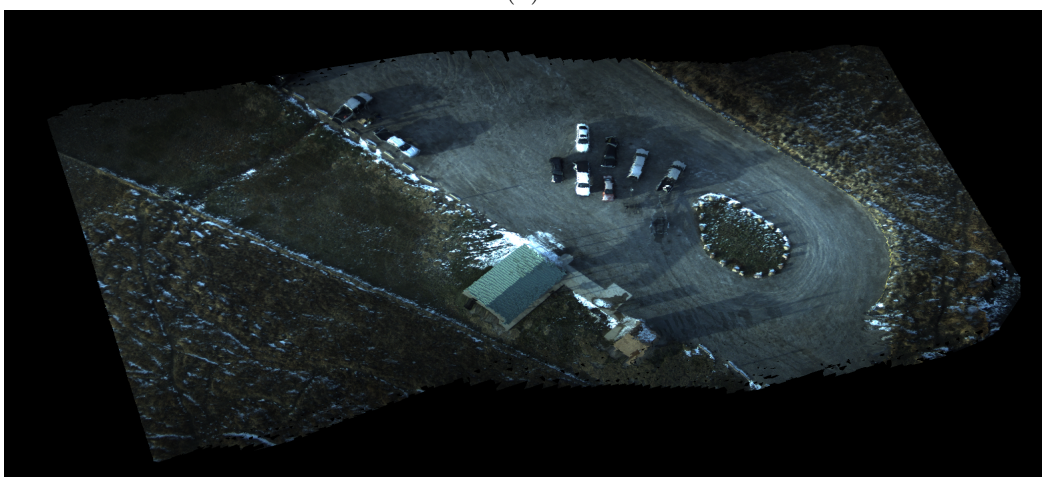
	SBA		**CBA**	
Number of texel images	**Time** (min)	**Memory** (GB)	**Time** (min)	**Memory** (GB)
30	3.75	5	7.68	7
50	7.30	6	13.11	13
75	9.33	6	20.26	23
100	12.53	6	32.55	35
150	18.25	6	68.43	63
200	24.38	6	failed	failed
300	33.36	6	failed	failed
500	46.28	6	failed	failed
1000	68.53	6	failed	failed



(a)

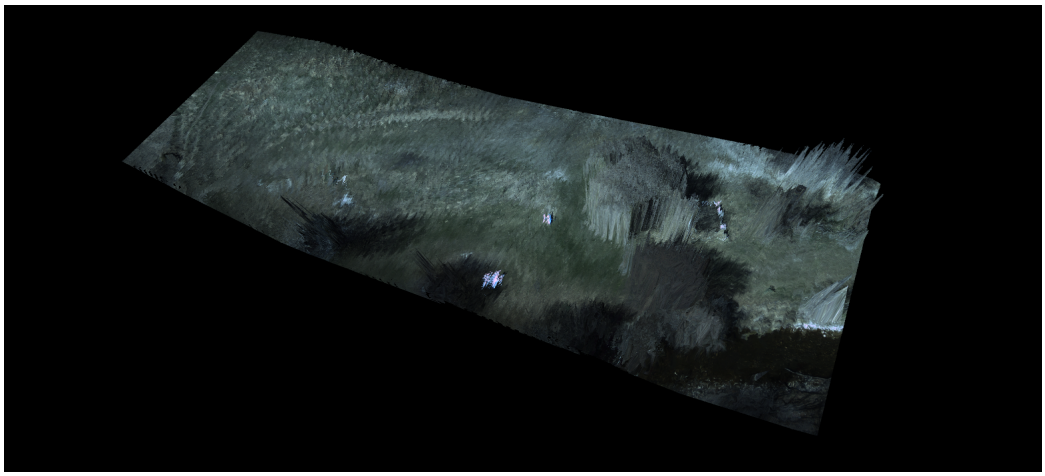


(b)

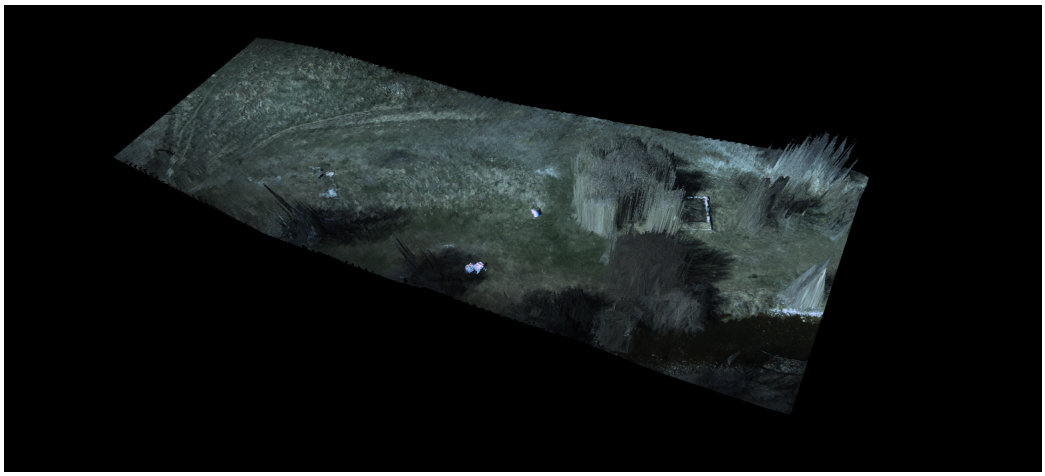


(c)

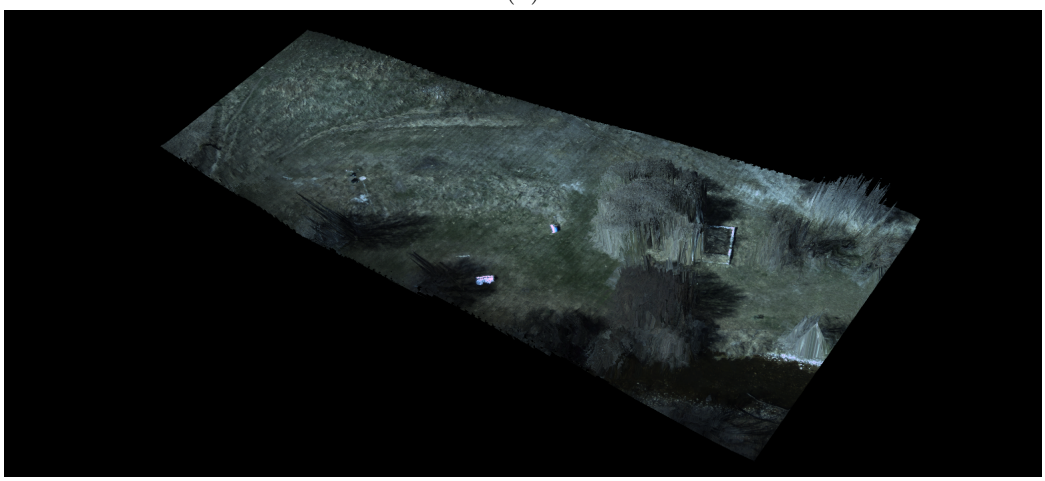
Fig. 3.4: Registration results (a) Before registration (b) Registration using CBA (c) Registration using SBA



(a)

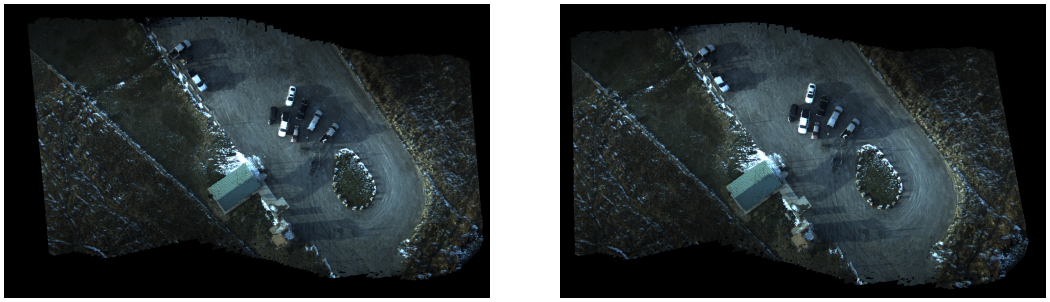


(b)

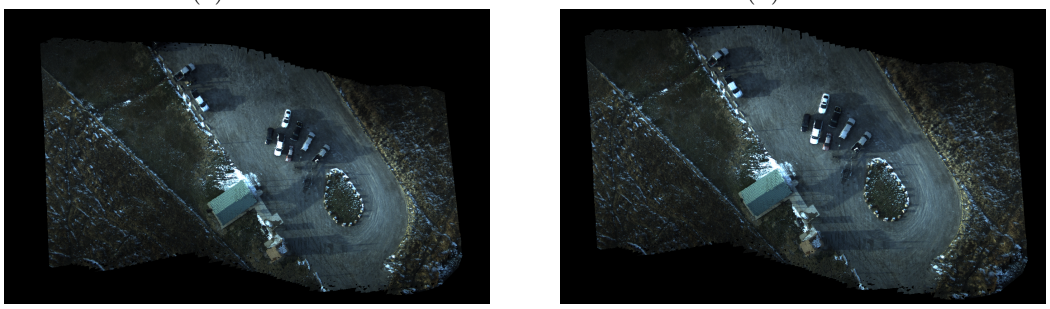


(c)

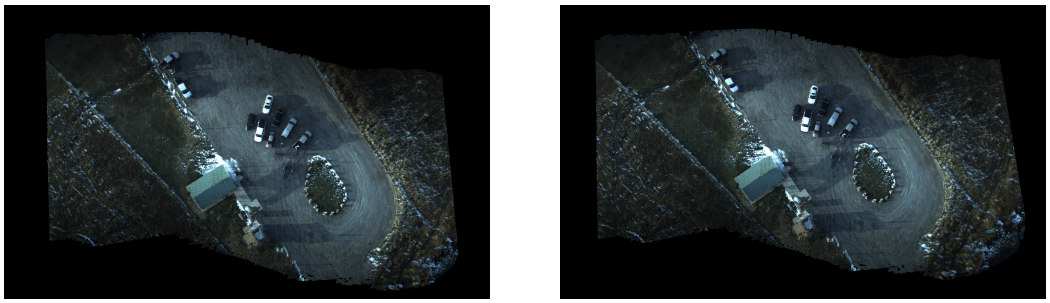
Fig. 3.5: Registration results (a) Before registration (b) Registration using CBA (c) Registration using SBA



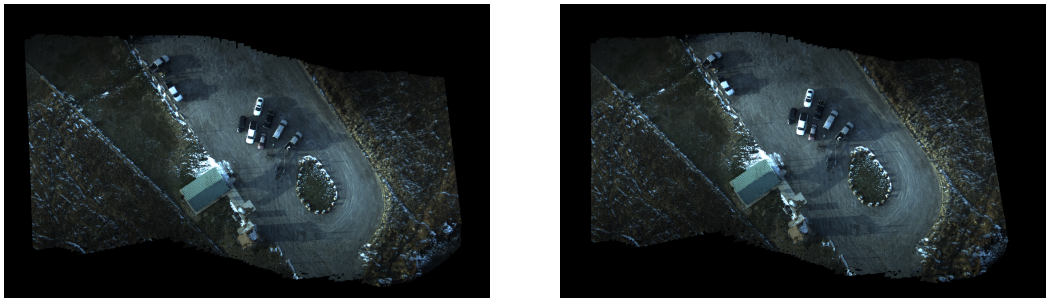
(a) $\mathcal{L} = 2$ (b) $\mathcal{L} = 3$



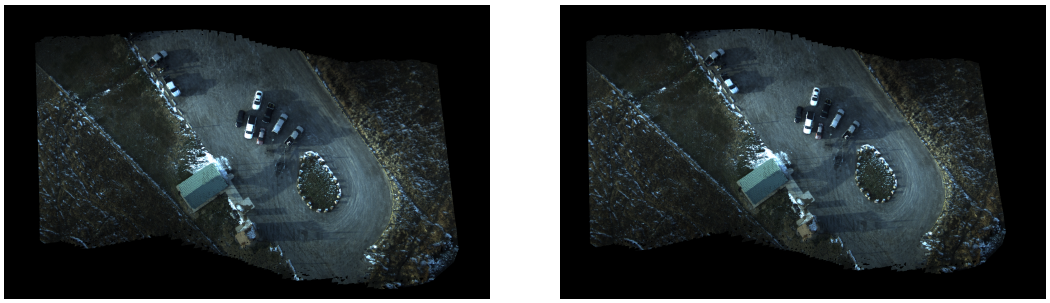
(c) $\mathcal{L} = 4$ (d) $\mathcal{L} = 6$



(e) $\mathcal{L} = 7$ (f) $\mathcal{L} = 8$

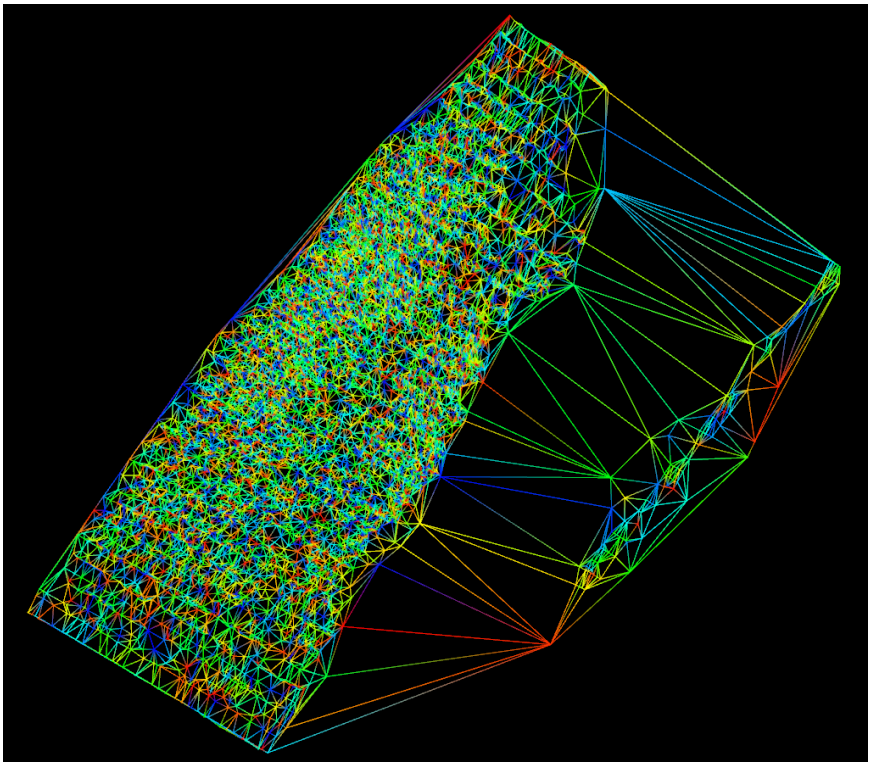


(g) $\mathcal{L} = 10$ (h) $\mathcal{L} = 12$

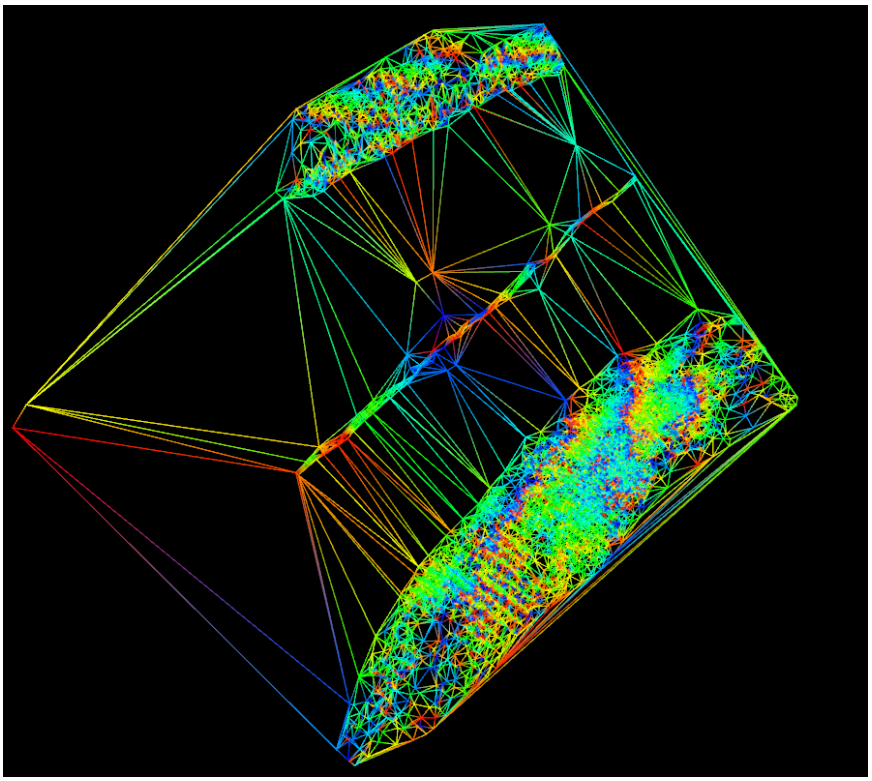


(i) $\mathcal{L} = 21$ (j) $\mathcal{L} = 28$

Fig. 3.6: TDSM with different *look length*



(a)



(b)

Fig. 3.7: Wireframe of the TDSM using SBA (a) TDSM 1 (b) TDSM 2

CHAPTER 4

Selection and Packing of Optimal Texture Fragments

4.1 Introduction

A texel camera comprises a scanning lidar sensor and a digital camera that generates fused lidar and image data called texel image. A texel camera also consists of an INS that gives a rough estimate of the attitude and position (pose) of each texel image. This camera is small enough to be mounted onto a Small Unmanned Aerial Vehicle (SUAV) and can be flown over terrain to gather texel images. The texel images are registered together, and the corresponding texture is computed to form a TDSM.

One of the fundamental steps in generating a TDSM is to generate appropriate texture, which is a compute-intensive process. The texture is saved in an image file and gives the visual information of the 3D scene. The texture for the 3D scene should be accurate, have the same resolution as the original 2D source image, and be seamless. Several methods have been proposed to determine the optimal texture for each triangle from the TIN, including a cost function that minimizes the angle between the normal of a triangle and the viewing direction of the input camera by Malik et al. [44]. Alshawabkeh proposed selecting the triangle with the maximum area after projection [45], while Hanusch proposed an angle-based method that selects the image with its optical axis normal to the mesh triangle [46]. El-Hakim et al. proposed a weighted blending of all available textures [47], which could be equally weighted or computed as a function of triangulated face projection resolution and viewing angle [48]. Other methods use 3D geometric criteria such as the angle between a triangle normal and the viewing frustum, distance to the camera, the surface area of a triangle face, and even photo-consistency metrics to map the proper texture [49, 50]. Alj et al. explored these methods and ranked them [51].

Packing the texture is needed to reduce the memory footprint of the resultant TDSM,

which is essential for a smooth viewing experience in the 3D Viewer. Frueh et al. packed each textured region by determining the best destination for each region [48]. The best destination for each region is determined by minimizing a cost function that depends on the distance of the destination to the top left corner of the atlas and the required graphics memory to place it. This works for a rectangular texture region like buildings but will suffer performance issues where the terrain surface results in irregular texture regions. Large-scale surveys in the real world are more likely to result in a large number of irregularly shaped textured regions where this packing will result in more empty spaces in the atlas [48]. Lai et al. packed the texture by forming a mesh-island for each view and employed the Oriented Boundary Box (OBB) method to construct a boundary box for each mesh-island [52]. These mesh-islands are packed together onto the UV map according to their OBB length on the UV map. However, the paper assumes a low possibility of getting non-uniform texture regions, resulting in very few mesh-islands. This has limited applications like e-commerce product presentation, as the author states. However, it is not suitable in applications of survey flight covering large regions as many irregularly shaped mesh-islands result in more space wasted in the packing process.

In the texel camera domain, the texturing is done on each pixel based on the nearest 3D point. It was assumed that for any output pixel, the best texture could be obtained from the texel swath containing the 3D point with a projection closest to the given pixel [31,32]. This method works for a lidar with a single beam and flat surface but presents problems when multiple-beam lidar is used or when the scene has a highly uneven surface. The final texture of the TDSM, when texturing the side of the object using the nadir viewpoint, results in texture with stretched pixels. For a multi-beam lidar, a visible seam in the texture is seen even in the flat area when lidar points from two separate beams in different texel swaths lie very close to each other. In the preliminary work, the problems associated with pixel-based texturing were shown, and a solution was discussed [53]. The use of view-dependent texturization overcame some problems that were discussed. After registering texel images, all the 3D points are projected onto a common image space called output image space. The

projected 3D points are triangulated in 2D space. The texel swath with minimum cross-track distance is selected as output image space. This was done to ensure the texture was downsampled instead of upsampling the textures. This resulted in the final texture losing some valuable resolution information. It would be better if the texture could be saved in its original resolution, which is the primary focus of the current work. Instead of transforming texture to the smallest swath space, the best texture for each triangle is found and saved in the original resolution. This also eliminates the error introduced by the texture transferring from the source swath space to the output swath space that was observed in the work [53]. The downside of the current method is that the output texture is non-contiguous and can have many image fragments. This warrants a proper texture packing algorithm to pack image fragments as tightly as possible in a single image canvas.

Section 4.2 introduces the registration of texel images. The algorithm for the selection of the best texture is discussed in Section 4.3, and Section 4.4 gives the overview of the texture packing algorithm. Finally, Section 4.5 shows the result of the algorithm.

4.2 Registration of Texel Images

The INS fitted onto the low-cost texel camera does not have a highly accurate measurement. The pose for each texel image computed using this measurement is subject to some instrument error. Furthermore, all the 3D points for each texel image calculated using the pose all suffer similar errors. If we are to create DSM with the measurement obtained directly from the texel camera, the texel images will not be registered properly. So, it is important to optimize poses and 3D points for each texel image to register them before we find the texture associated with the final DSM.

Registration involves finding the common lidar projection points onto all neighbor texel images. The final optimization is done by forming a cost function that involves lidar range measurement, their calibrated projection onto the associated texel images and projection onto neighboring texel images. This cost function is minimized using the LM algorithm to optimize poses and all the 3D points for each texel image. The method of registering texel images with a single beam is described in work by Bybee and Budge [32], and Khatiwada

and Budge extended it to texel images with multiple beams [31].

4.3 Texture Selection

This section discusses the selection of a texture for any particular triangle from possible contenders. In the last section, we registered all texel images in a common frame and optimized the poses and 3D points for each texel image as well. These 3D points are then triangulated using Delaunay triangulation to create a TIN. Each triangle in this TIN is referred to as the 3D triangle in this document. The next step is determining the texture for final surface maps using the best texture available. The best view for the final model can be calculated in two ways. The first way is to save complete texture from all texel images in the final TDSM and select the best view when viewed in a 3D Viewer. Another way is to construct a single image by finding the best possible texture for each triangle in the image. Even though the first way potentially delivers a better view, the amount of storage needed to save the texture for all images is huge. It is computationally prohibitive because every time the view in the 3D Viewer is changed, the best texture for all the triangles must be calculated again. The second method eliminates redundancy, is much more economical in data storage, and still results in the best possible view.

To find the best view for a given 3D triangle, all three vertices of the 3D triangle are projected back to each image using the optimized pose we got from texel image registration as described in the previous section and the mapping coefficients obtained from the calibration of the Texel Camera. If the projected triangle falls within an image boundary, then the camera can see the 3D triangle. We repeat this process for all triangles to obtain candidate images for them. The best possible texture for a triangle is obtained from the best candidate image. The following criteria are then used to select the best texture (or candidate image) for each triangle.

- **Resolution score**: Each 3D triangle is first projected onto all the candidate images. The area of the projected triangle is calculated in each case. A triangle with a bigger area is preferred as it corresponds to a higher number of pixels. Thus, the large area

of the projected triangle has a better texture resolution compared to the smaller area. The Resolution score is normalized to fall within the range of $[0, 1]$ and is given by

$$R_i = \frac{A_{\triangle_i}}{\max(A_{\triangle})}, \tag{4.1}$$

where $A_{\triangle_i}$ is the area of the projected triangle in the i^{th} candidate image and $max(A_{\triangle})$ is the maximum area of the projected triangle.

- **Distance score**: The distance between a 3D triangle and a candidate image plays an important factor in determining the best image. A closer distance is preferred. After registration of texel images, the COP of each image can be obtained from the optimized pose. The Distance score for each candidate image can be determined using a Euclidean distance formula. The Distance score is also normalized to fall within the range of $[0, 1]$ and is given by

$$D_i = 1 - \frac{D_{\triangle_i}}{\max(D_{\triangle})}, \tag{4.2}$$

where $D_{\triangle_i}$ is the Euclidean distance from center of the 3D triangle to the i^{th} camera's COP.

- **Angle score**: To get the best texture, the angle between the triangle and the view vector is desired to be as small as possible. The normal ($\vec{n}$) is calculated from the triangle vertices, and the view vector ($\vec{v}$) is calculated from the COP of the image and the center of the triangle. We calculate the angle between $\vec{n}$ and $-\vec{v}$ by

$$\theta = \cos^{-1}\left(\frac{-\vec{v}.\vec{n}}{|\vec{v}||\vec{n}|}\right).$$

If the angle lies in the range $[-90, 90]$, the Angle score is computed by finding the cosine of that angle. The Angle score is set to zero for any angle outside the range and is computed as

$$A_i = \begin{cases} \cos(\theta), & \text{if -90} < \theta < +90 \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

- **Orthoimage score**: The images captured by the camera suffer the least distortion towards the center of the image. To get the best view, a view from the middle region of the candidate image is preferred rather than toward the edges of the candidate image, as it is an ortho-view to the triangle. If $(r_\triangle, c_\triangle)$ is the center of the triangle and (r_i, c_i) is the center of i^{th} image, then the Orthoimage score is calculated as

$$O_i = 1 - \frac{\sqrt{(r_i - r_\triangle)^2 + (c_i - c_\triangle)^2}}{(Dg_i/2)}, \tag{4.4}$$

  where Dg_i is the length of the diagonal of i^{th} image.

After calculating four metrics for each candidate image, the total score is then computed as follows:

$$S_i = \frac{w_1^s R_i + w_2^s D_i + w_3^s A_i + w_4^s O_i}{w_1^s + w_2^s + w_3^s + w_4^s}. \tag{4.5}$$

Here, $w_1^s, w_2^s, w_3^s, w_4^s$ are the texture score weights for the Resolution score, the Distance score, the Angle score, and the Orthoimage score, respectively. These weights determine the significance given to each score. In this study, more significance is given to the Resolution score and the Angle score, and thus, they are given higher weights. The weights chosen in this study are $w_1^s = 4$, $w_2^s = 1$, $w_3^s = 4$, and $w_4^s = 1$

For each triangle, the candidate image with the largest S_i is chosen as the owner of that particular 3D triangle. This owner is used when the triangle needs to be textured while creating the output texture. This process is repeated until all the triangles are assigned an owner.

- **Neighbor coherence**: After determining the owner for every triangle, we check the neighbor of each triangle. If all three neighbors favor the same owner, but that owner

is different from the current owner, then the current triangle's owner is changed to the owner favored by the neighbor to maintain coherence. The triangle must be visible from the new owner to complete the change. This results in seamless textures between those neighbors.

- **Filling holes**: If all three vertices of a triangle are not visible in any image, the candidate image list for that triangle would be empty, and no owner would be assigned to it. This creates a "hole" in the final texture. Since we can't fill this triangle with the texture from a single image, we have to use the closest 3D point criterion to fill part of the triangle. For each pixel inside this triangle, we find the nearest 3D point and transform the texture from the owner of that 3D point to the output image space, as described in previous work [32].

4.4 Texture Packing

After finding each triangle's owner, the texture for each triangle needs to be saved in a single texture file. Since packing triangles in a rectangle is an NP-hard problem, we convert them into rectangles by grouping them based on the criteria explained below. Then, those rectangles are packed together into a bigger rectangle called a canvas. Texture packing proceeds in two stages:

4.4.1 Grouping triangles into rectangles

If a rectangle is formed from each triangle by enclosing the triangle in a bounding box, then the rectangle is at least 50% empty, which is not desirable when trying to pack triangles densely. So we group contiguous triangles such that contiguous textures could be enclosed to form one rectangle with less empty space. To determine if two triangles are contiguous, we use the owner information we obtained in the last section.

The triangles are packed in a Breadth First Search (BFS) fashion. The starting state is just one triangle. The packing score of a bounding box is computed for each packing state. We then inspect the list of neighbors of the starting triangle to determine which can

be grouped together. Two triangles can be grouped if they share the same owner image. Each addition to the group forms a new state. Similarly, in the next stage, we inspect the neighbors of all the triangles that were added to the group. This continues until we run out of triangles to add to the group.

After grouping the triangles, we use the texture from the state with the best packing score. The BFS grouping of triangles is repeated for all the triangles that are not in the best packing state. The packing score and all metrics used to compute the packing score are defined below:

- **Shape score**: To pack compactly, it is desirable that the shape of the bounding box be as square as possible when there are an arbitrary number of bounding boxes of arbitrary shapes possible. A mix of skinny horizontal and vertical bounding boxes makes it difficult to pack the rectangles densely. The Shape score is defined as

$$Sh_i = 1 - \frac{(h_i - b_i)^2}{\max(h_i, b_i)^2}, \tag{4.6}$$

  where h_i and b_i are the height and width of the bounding box at state i.

- **Size score**: A rectangle formed by enclosing a triangle in a bounding box has at least 50% wasted space. The wasted space can be potentially reduced by grouping two or more triangles such that the grouped triangles form a rectangle-like shape. In this paper, large groups were preferred because a large group implies a larger contiguous texture, resulting in fewer seams when stitching and lower wasted spacing when packing. The Size score is defined as

$$Sz_i = \frac{N_i}{(N_i + k)}, \tag{4.7}$$

  where N_i is the number of triangles at state i and k is an empirical constant to adjust preferences for group size. The larger the k, the higher the preference for large grouping.

- **Occupancy Score**: It is defined as the ratio of the total pixels inside a bounding box to the area of the bounding box. The higher the occupancy score, the lower the empty space it has. Rectangles with higher occupancy score result in dense packing. The Occupancy Score is defined as

$$Oc_i = \frac{Px_i}{b_i \times h_i}, \tag{4.8}$$

  where Px_i is the number of pixels inside this rectangle, and h_i and b_i are the height and width of the bounding box at state i.

- **Packing Score**: The Packing score is computed as the weighted sum of the Shape score, the Occupancy score and the Size score. Mathematically, it is defined as

$$P_i = \frac{w_1^p Sh_i + w_2^p Sz_i + w_3^p Oc_i}{w_1^p + w_2^p + w_3^p}, \tag{4.9}$$

  where w_1^p, w_2^p, w_3^p are respective packing weights. Similar to the texture selection weights, these packing weights determine the significance given to each score and can be chosen ad hoc. Since dense packing is highly valued in this study, the Occupancy score is given more weight when calculating the Packing score. The weights chosen in this study are $w_1^p = 1$, $w_2^p = 2$, and $w_3^p = 7$

At the end of grouping triangles using BFS, all of the triangles are grouped and enclosed inside a rectangle. Each group may have one or more triangles.

4.4.2 Packing rectangle into a canvas(output texture)

The problem is to pack N rectangles as densely as possible into a canvas. There are many ways to sort the rectangles before packing. The packing algorithm's performance depends on the type of sorting and the nature of the shape and size of the rectangles. In this paper, the rectangles are sorted according to their width.

To pack rectangles, a rectangle is placed in the top-left corner of the canvas. This divides the canvas into two empty regions, as shown in Fig 4.1. Similarly, the remaining

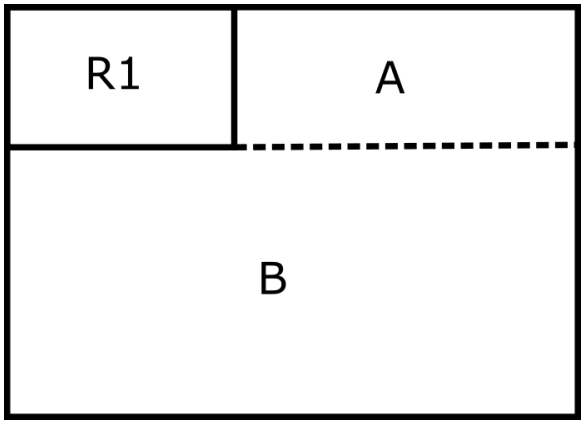


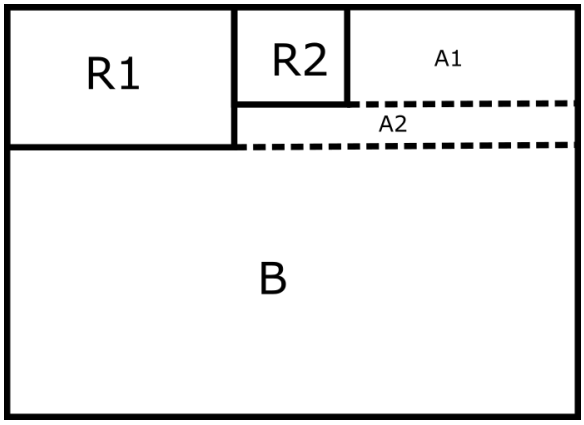
Fig. 4.1: Canvas after placing first rectangle, R1

rectangles can be placed in the top-left corner of either empty region where they fit. This further splits that region into two more empty regions, which can take the form of either Fig 4.2(a) or Fig 4.2(b). This process is repeated for all the remaining rectangles placing them appropriately in a binary tree fashion. We refer to this packing algorithm as the base packing algorithm. The base packing algorithm takes a list of input rectangles, maximum height and maximum width and returns the resulting height and width of the canvas after packing.

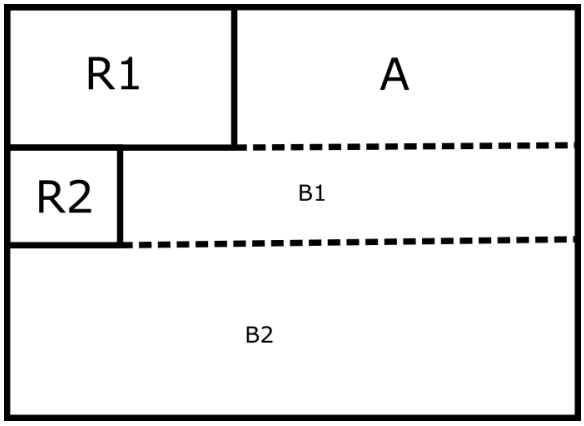
While the base packing algorithm results in a decently packed canvas, it also leaves a lot of empty space in the bottom-right region of the canvas. We devised an iterative method to reclaim that empty space to increase packing density further. First, the base packing algorithm calculates an initial height and width. Then the rectangles are repacked in an iterative fashion. The original canvas is divided into three regions, as shown in Fig 4.3. The base packing algorithm is run on Region I. If there are unpacked rectangles, the packing algorithm is run on Region II, and the same for Region III.

After all the rectangles are placed in this canvas, we compute the height, H_1 and the width, W_1, for the resulting canvas as:

$$\begin{aligned} H_1 &= h(I) + \max[h(II), h(III)] \\ W_1 &= \max[w(I), w(II) + w(III)] \end{aligned}, \tag{4.10}$$



(a) When placed in area A



(b) When placed in area B

Fig. 4.2: Second rectangle,R2, can be placed in either region A or B from Fig. 4.1

where $h()$ is the height and $w()$ is the width of the respective region.

For the next iteration, the canvas height is reduced by a factor of 0.95, and the process is repeated for a certain number of iterations or until the rectangles can no longer be packed in the given canvas.

4.5 Result and Discussion

The algorithms described in this chapter are run on the same Cutler data that was used in Chapter 3 and the entire TDSM generation algorithm is executed. Both kinds of textures, the legacy way with the nearest 3D point, and the newer way of finding the optimal triangle, are generated. When the texture is generated using the optimal triangle method, a packing algorithm is implemented to pack all the triangles into a single image

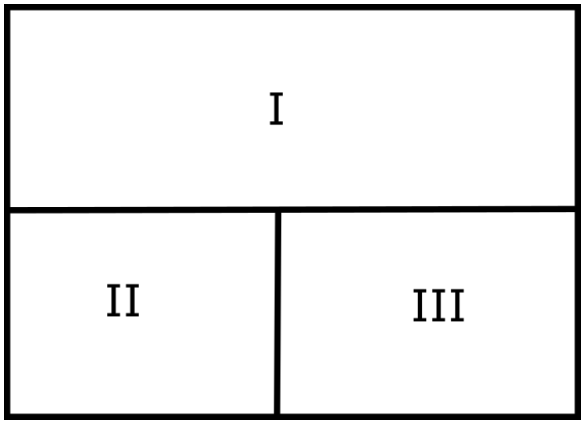


Fig. 4.3: Canvas is divided into three regions, I, II, and III.

canvas. Since the algorithm is tested on real data and not simulated data, it is harder to quantify the result. One metric for analyzing the effectiveness of the new texture is by counting the number of pixels in the texture for both the regular method and the optimum triangle method. The other metric is a visual metric where a comparison is made between the previous method and the optimum texture method. The visual comparison is made by generating the TDSM and looking at the quality of texture between each method. Fig 4.4 shows the point cloud after the triangulation is done forming a TIN. From this wireframe image, it can be seen that each triangle in the TIN has 3D coordinates in world space, and an orientation of the triangular surface. The texture selection algorithm is run for each triangle in the TIN and the appropriate score is calculated. The texture with the highest score is assigned as the candidate image for a particular triangle. Fig 4.5 shows the output image canvas after selecting the best texture triangle for each 3D triangle in a TIN. These triangles are packed using the algorithm described in Section 4.4. By comparison, Fig 4.6 shows the regular texture created using the existing method.

Since it is almost impossible to compare the regular texture shown in Fig 4.6 and optimal texture triangles in Fig 4.5, a TDSM is constructed with both textures. The current 3D Viewer software was modified to read in the texture shown in Fig 4.5 and the user can choose which texture to use during the visualization stage. Fig 4.7 shows the TDSM with both regular texture and the optimal triangle texture. On closer inspection,

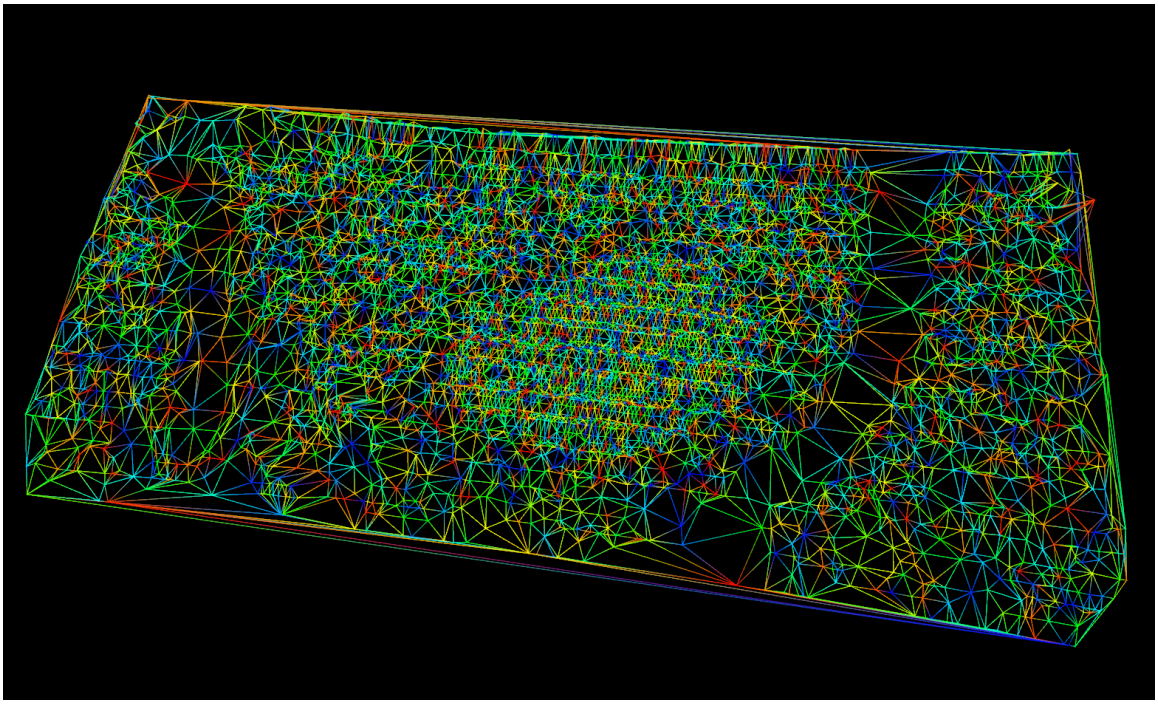


Fig. 4.4: Wireframe of the TIN

the texture generated with optimal triangle is slightly better than the regular method. Another comparison is shown in Fig 4.8 where a slight improvement is seen. Fig 4.9 shows some major advantage of using triangle method to the existing method.

A second dataset was taken by mounting texel camera onto a rotorcraft and flew over various buildings on the Utah State University. A single TDSM is created using 21 texel images. Fig 4.10(a) shows the unregistered point cloud obtained from given texel images and Fig 4.10(b) shows the point cloud after the registration algorithm described in Chapter 3 was applied. Like before, these data points in point cloud are triangulated to form a TIN and appropriate texture was selected based on all available textures. A comparison was made with the existing texturing method and shown in Fig 4.11. The result by using the texturing method described in this chapter is remarkably better than the existing one. The texture using the new texturing method is more smooth with less distortion and provide better visual quality as evident in the sidewalks area of the TDSM.

Table 4.1 shows the total number of pixels in the texture file for the output TDSM

Fig. 4.5: Texture with optimal triangles packed with the packing algorithm

between the existing method and the new method. In all cases, the number of pixels in the new texture is more than the texture using the existing method. The increment percentage is more as the number of images in a single TDSM gets higher. The only anomaly result when the increase in percentage is not significant is when the TDSM contains very flat region with a very little elevation change.

Fig 4.12 shows the texture using new method for a TDSM where 21 images are used to create it. The packing algorithm as described in Section 4.4 was applied before generating the texture. Fig 4.12 was zoomed to show details in Fig 4.13. Figure 4.13(a) shows the top half of the canvas where more triangles are packed in a boundingbox where Figure 4.13(b) shows the bottom half and mostly contains individual triangles.

Table 4.2 shows the result for the packing algorithm. The initial size column represents

Fig. 4.6: Regular texture

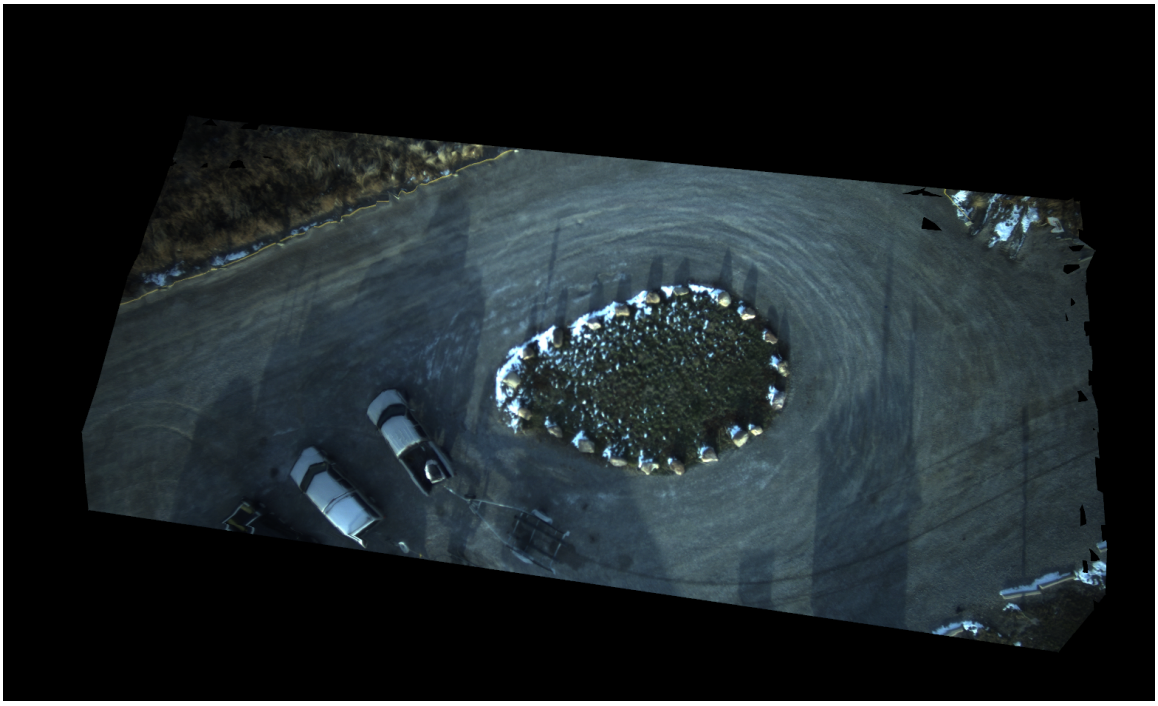
the size of the final image canvas if all the available images are saved. The next column shows the size of the image canvas when just the Base packing algorithm is applied. Finally, the last column shows the image size obtained when the iterative algorithm is applied after splittling the image canvas into three regions as shown in Fig 4.3

Table 4.1: Table showing number of pixel in existing method and the

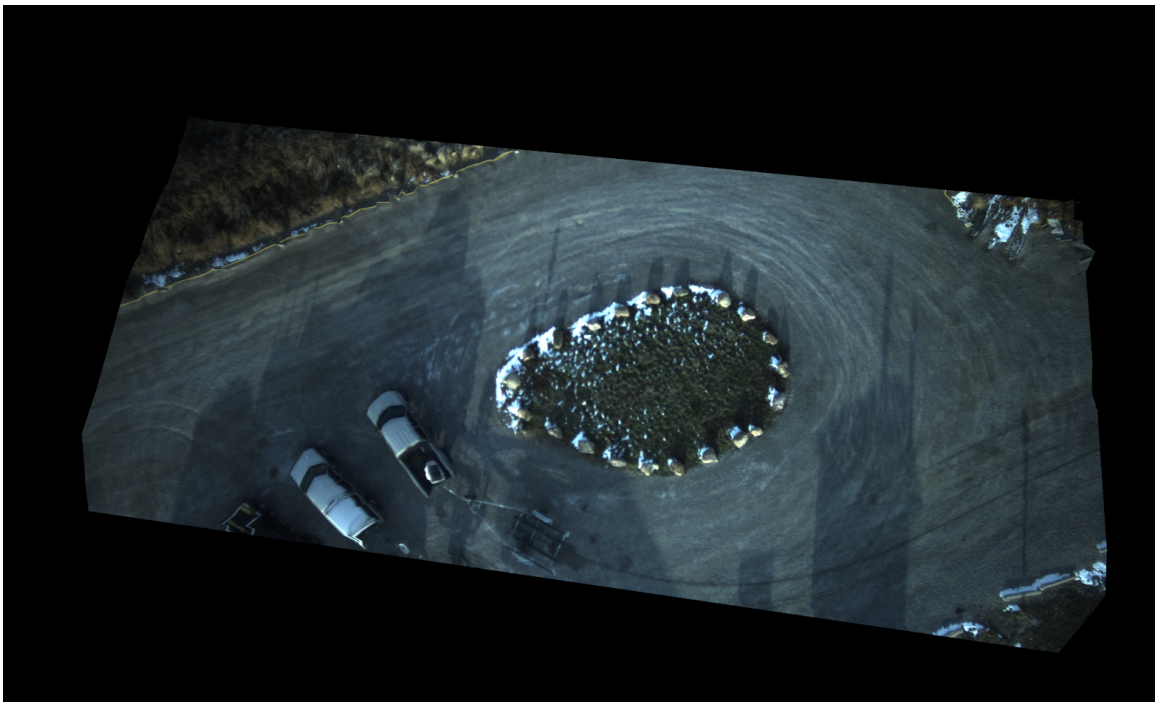
	Number of pixels		
Images in a TDSM	Existing method	New method	Pixels increase%
5	8010167	8067146	0.7
7	8444989	8746671	3.57
10	9378004	10012041	6.7
15	11248774	11844931	5.29
21	13915786	15314213	10.04
21	12644823	14335600	13.37
21 (flat region)	12006850	12206440	1.66
21	21751884	26104498	20.01
21	17137159	20311758	18.52

Table 4.2: Comparison of the final image size between initial, base packing, and splitting canvas with iterative algorithm is applied

	Image size in pixel (Row, Column)		
Images in a TDSM	Initial	Base packing	Final
5	(8800, 4240)	(8086, 4240)	(4584, 4240)
7	(12320, 4240)	(8770, 4240)	(5396, 4240)
10	(17600, 4240)	(10079, 4240)	(6677, 4240)
15	(26400, 4240)	(12103, 4240)	(8014, 4240)
21	(36960, 4240)	(12663, 4240)	(9290, 4240)
21	(36960, 4240)	(14241, 4240)	(10458, 4240)
21	(36960, 4240)	(12816, 4240)	(8936, 4240)

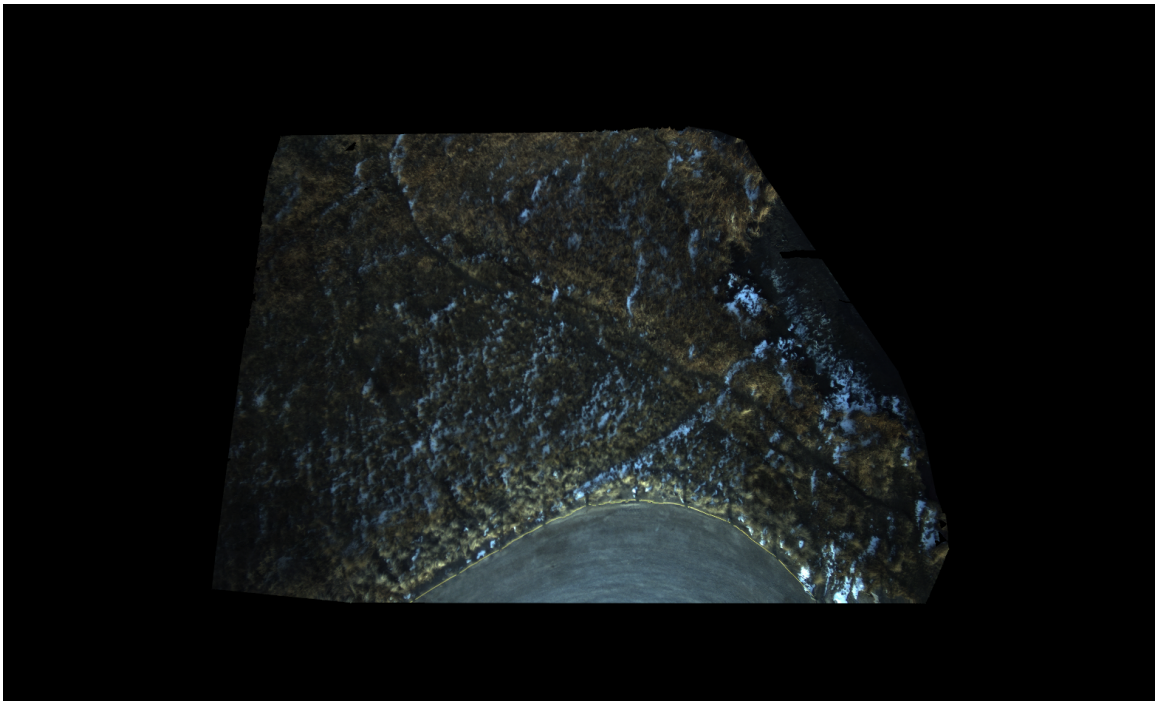


(a) Regular Texture

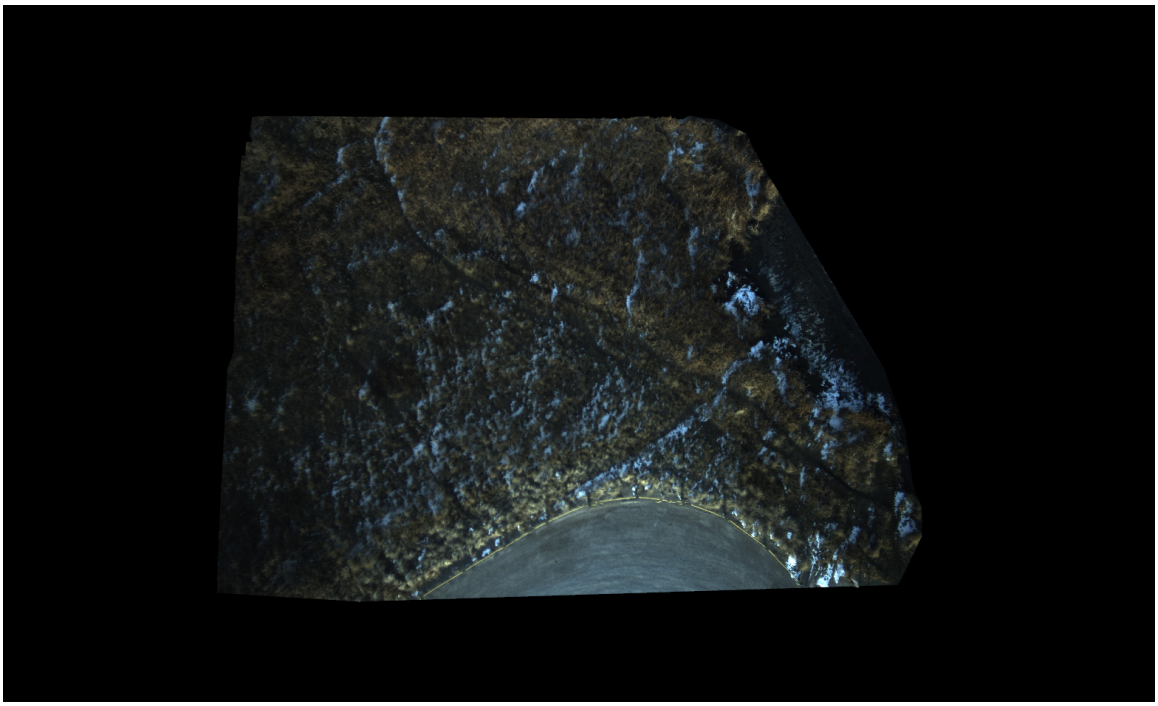


(b) Optimal triangle Texture

Fig. 4.7: Registered output TDSM

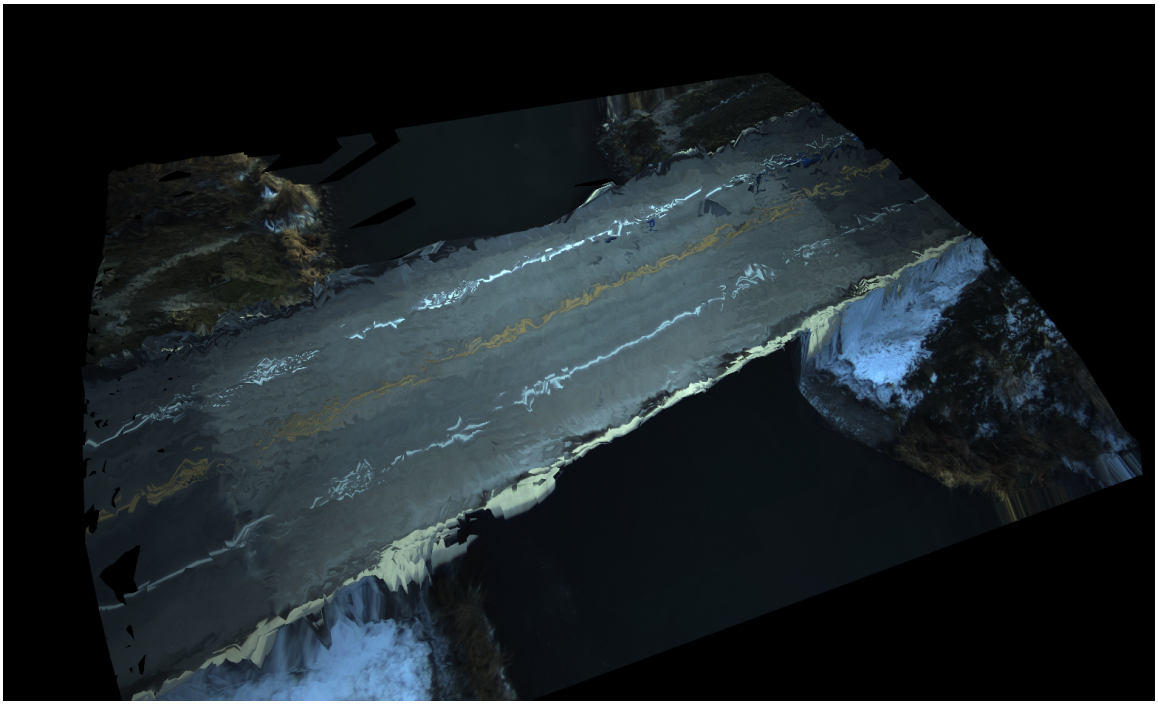


(a) Regular Texture

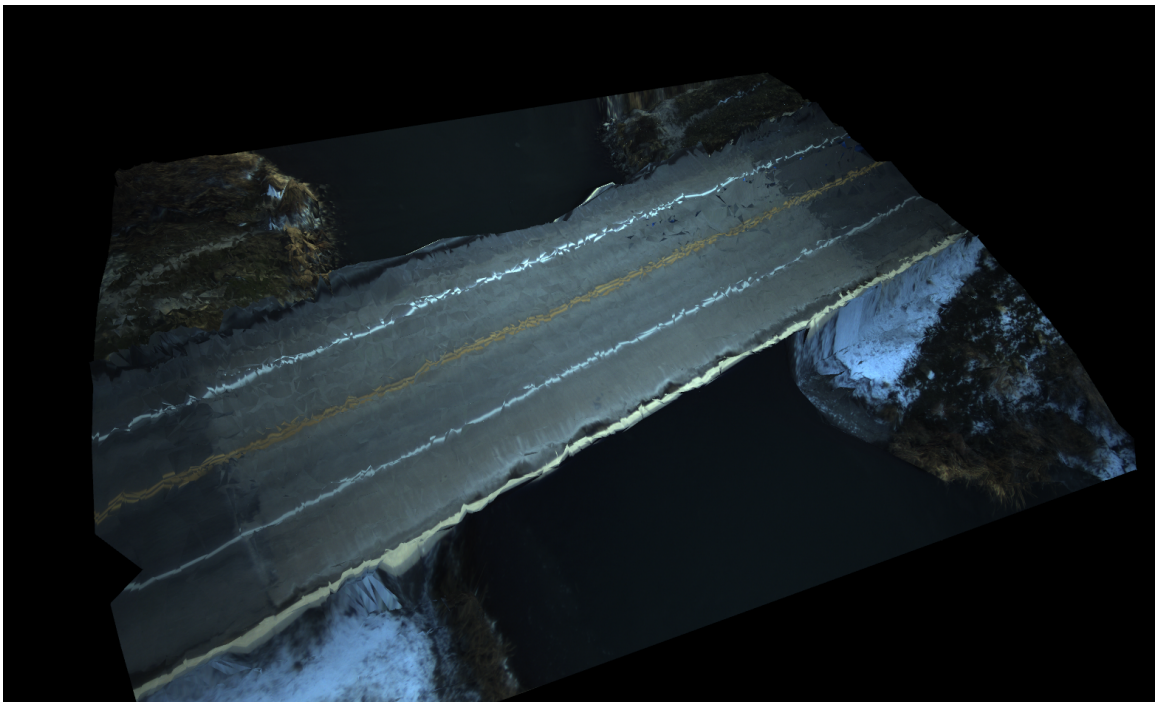


(b) Optimal triangle Texture

Fig. 4.8: Registered output TDSM

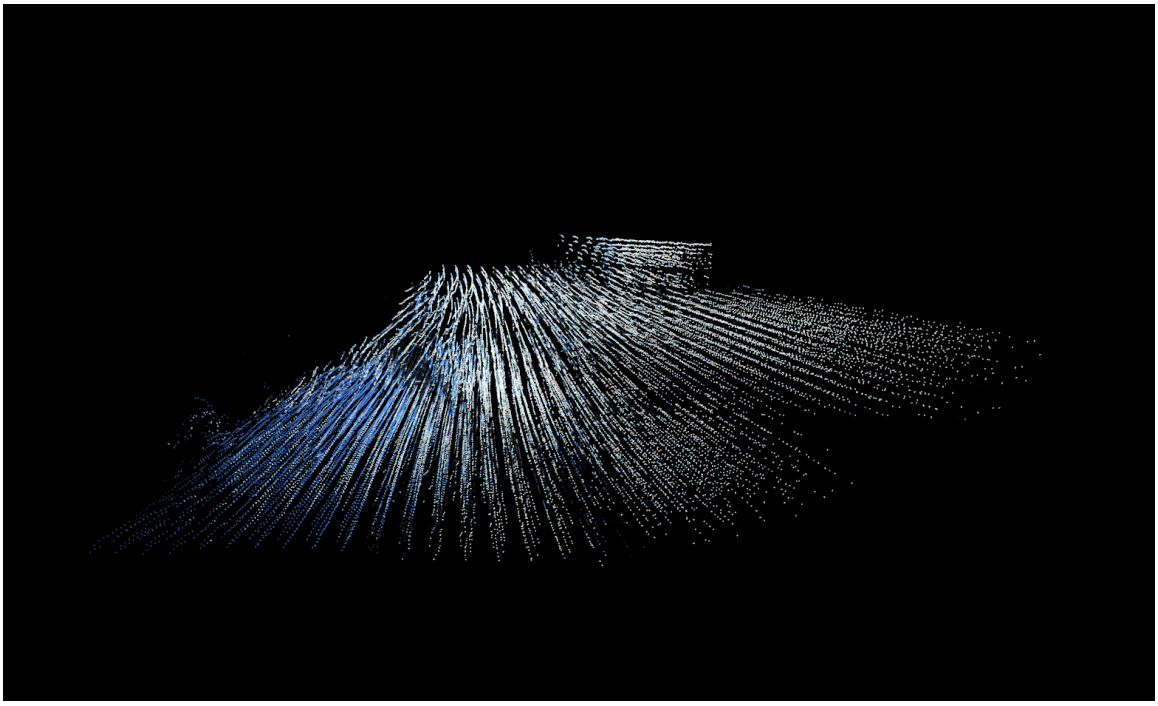


(a) Regular Texture

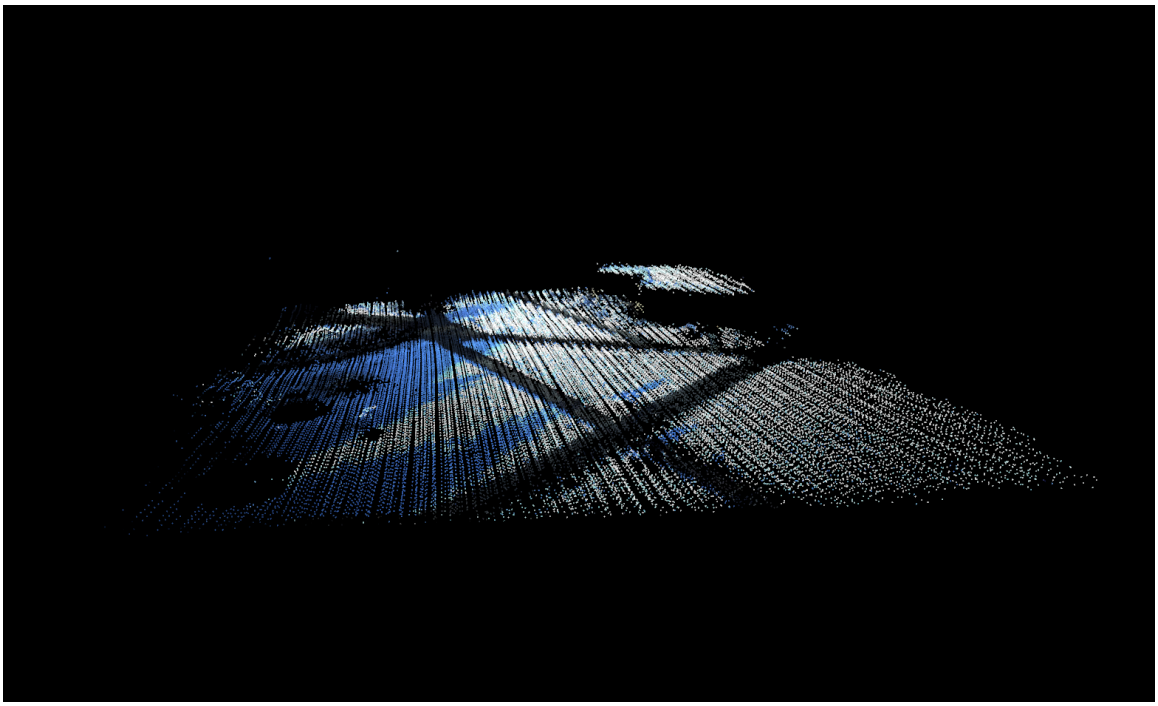


(b) Optimal triangle Texture

Fig. 4.9: Registered output TDSM

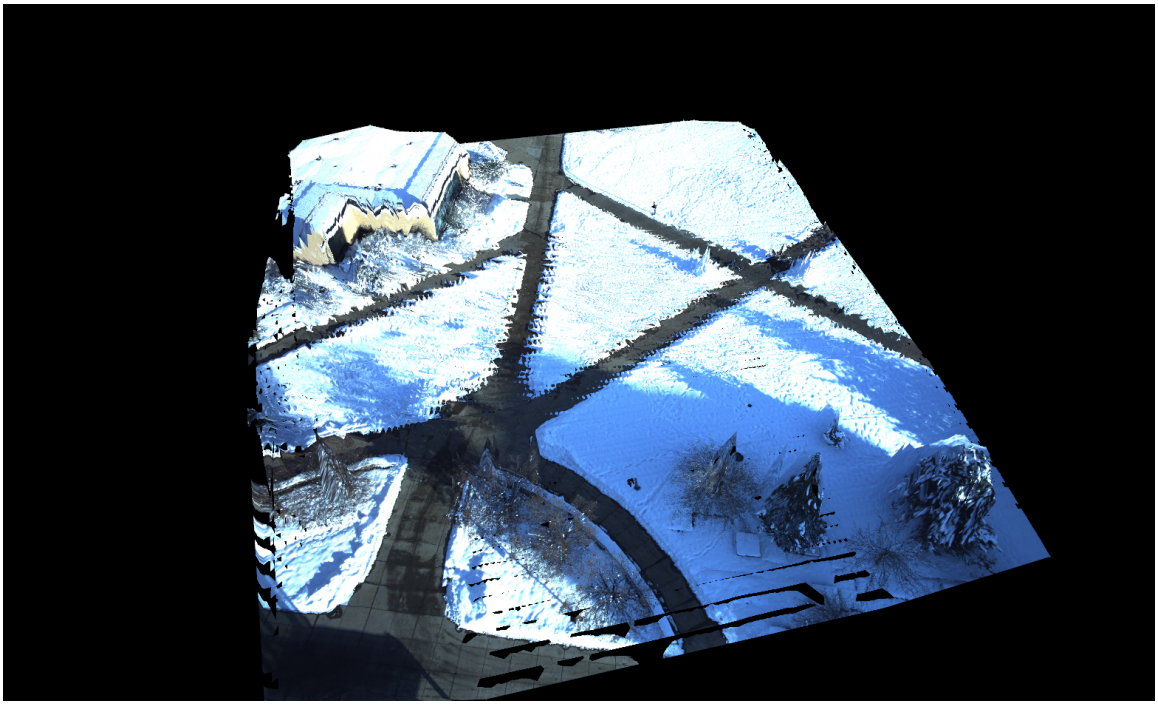


(a) Unregistered

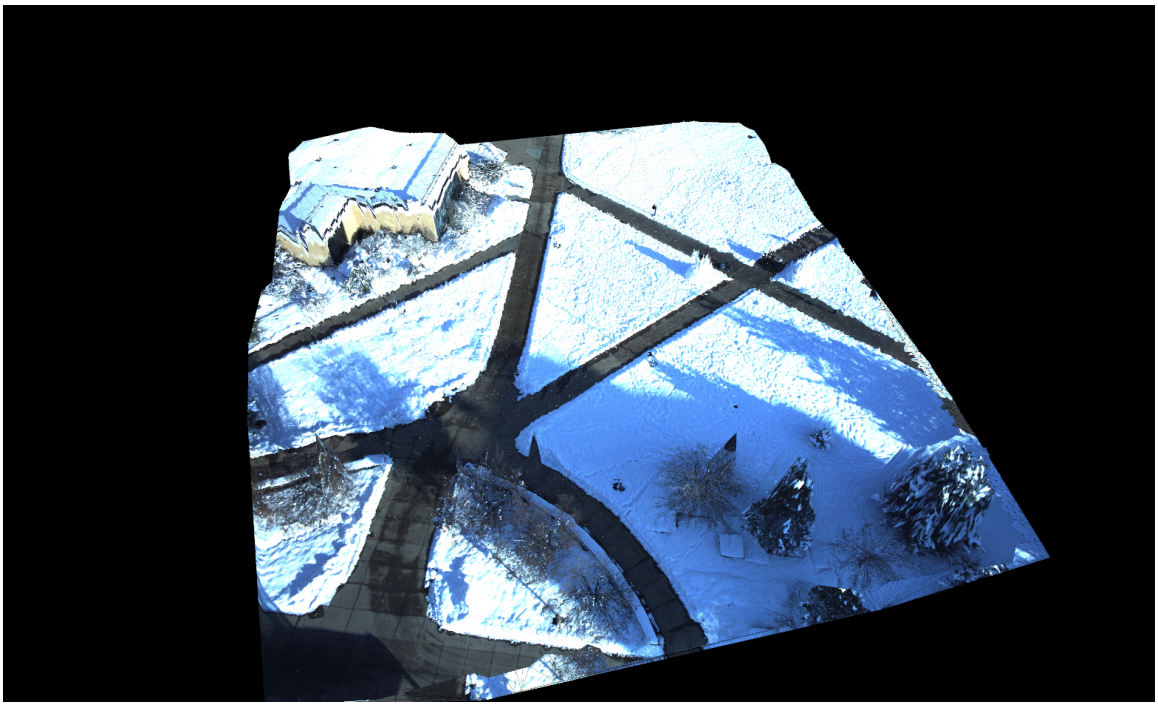


(b) Registered

Fig. 4.10: Textured Pointcloud



(a) Regular Texture



(b) Optimal triangle Texture

Fig. 4.11: Registered output TDSM

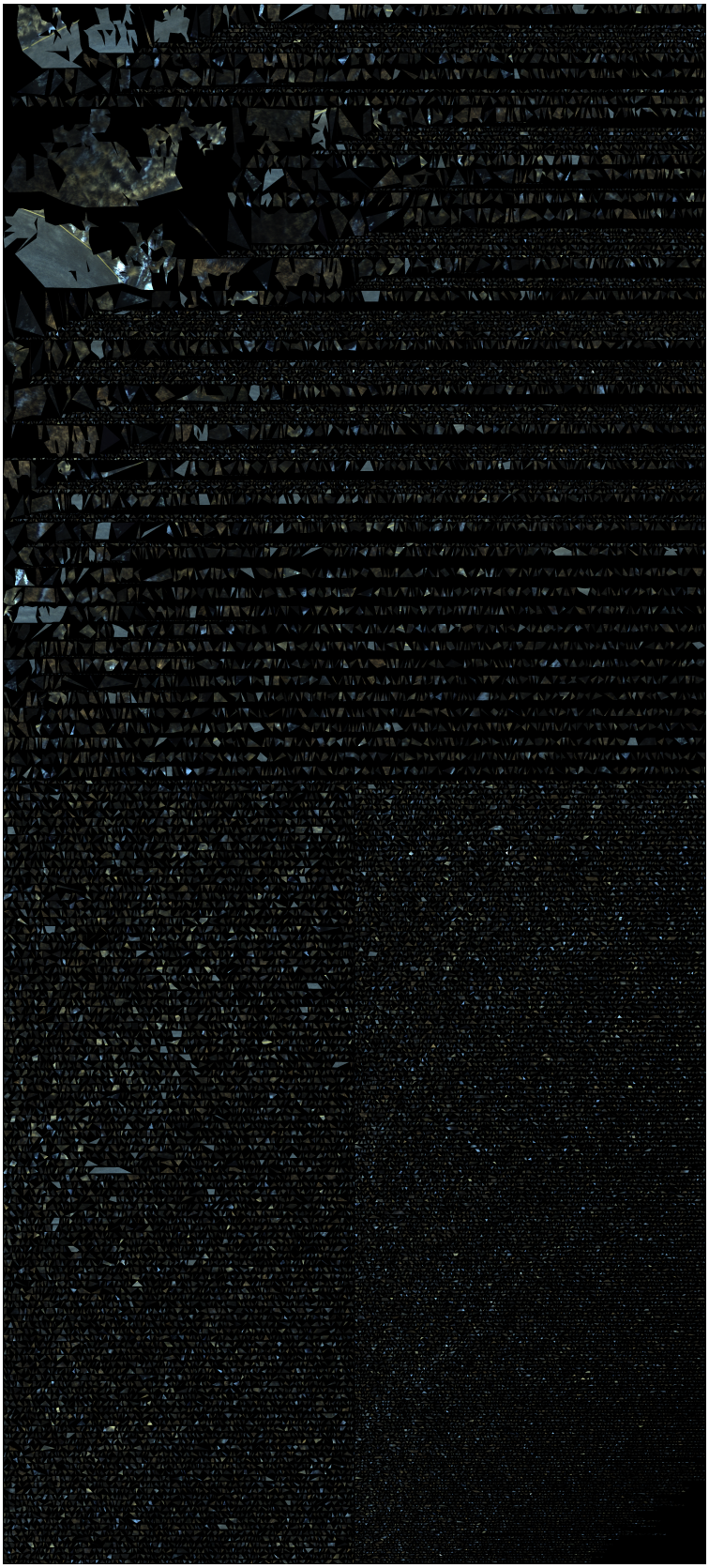
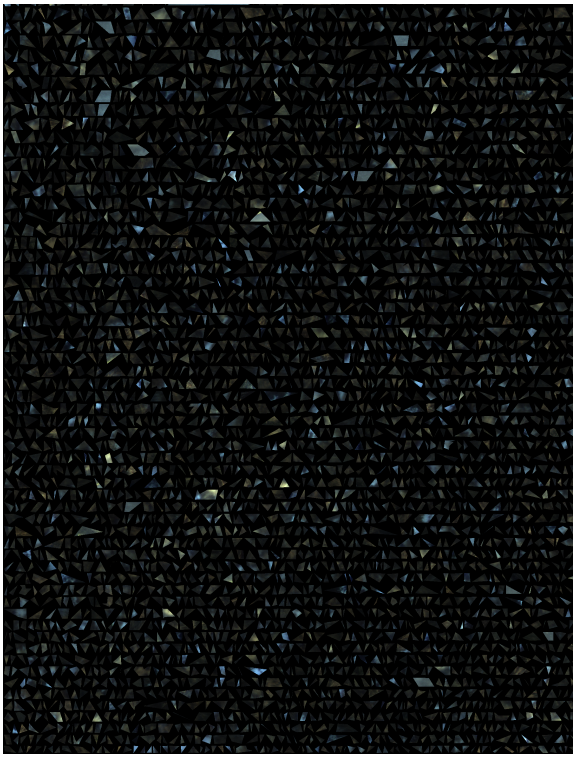
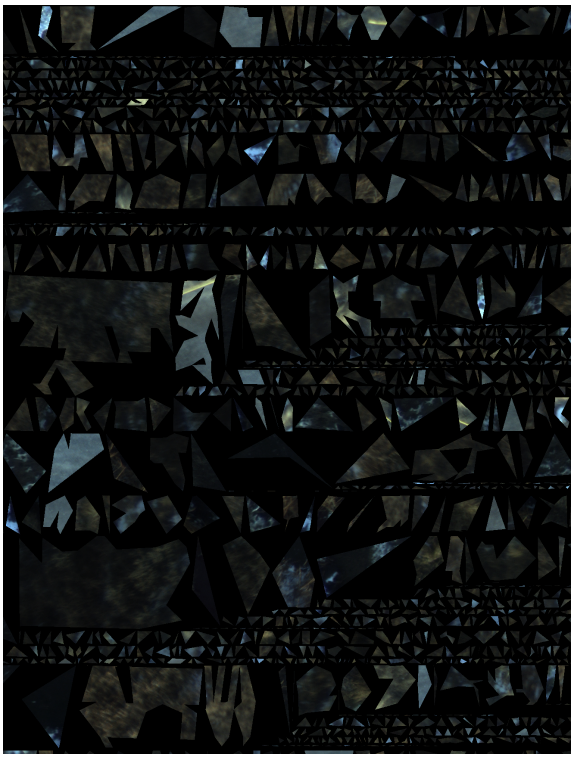


Fig. 4.12: Output image canvas after packing algorithm



(a) Zoomed on the top half



(b) Zoomed on the bottom half

Fig. 4.13: Zoomed to show details

CHAPTER 5

Superresolution

The resolution of an image is determined by the amount of detail it carries. The higher the resolution, the more details an image can display. A high resolution image can help distinguish between two separate features that are located close to each other in physical space. High-resolution images are usually desired in most applications as they are more pleasant to view and are more valuable for further processing and analysis. A digital image is comprised of pixels, which are the smallest addressable elements. A higher number of pixels in an image usually results in high resolution image, given that each pixel holds unique information.

The obvious way to increase the resolution of the image is to increase the number of sensors in the camera's sensor array. The hardware for the optical instrument to achieve such high resolution can be very costly and is not feasible for small commercial and scientific applications. Another approach is to reduce the size of the sensor array. However, this means the amount of light entering the sensor decreases, resulting in a poor image.

A more common and practical method to increase the image's resolution is to use a signal processing technique in the post-processing stage. This approach to increasing the resolution of digital images is called Superresolution (SR). This technique restores low-resolution images into clear, high-resolution images. Instead of enhancing the hardware, we can use SR techniques to go beyond the capabilities of low-resolution (LR) observations. Over the last three decades, many algorithms have been proposed to do SR [54]. SR technique can be applied to various fields like satellite imaging [55], video enhancement and restoration [56], medical imaging [57], remote sensing applications [58] and many more. Although the idea of SR was first addressed by Huang and Tsay [59], the term superresolution was mentioned first in 1990 by Irani and Peleg [60]. Since then, many applications have been formed based on the concept of the Superresolution technique [55, 61, 62].

SR can be broadly classified into two categories: multi-frame and single-frame. A learning method is used in a single-frame SR to upscale a single low-resolution image to a high-resolution image. Such a method is also called Example-based Superresolution [63]. In such a method, a prior is learned on the original high-resolution image using a database of numerous other images. The superresolution is performed by the Nearest-Neighbor(NN)-based estimation by learning correspondence between pair of low and high-resolution image patches from a database [64]. Others have tried to improve the NN-based estimation by including a regularized framework to overcome the overfitting problem commonly seen in NN-based estimates [65, 66]. He and Siu use Gaussian process regression to predict the neighbor pixel to improve the resolution without using an external training set [67]. Kim and Kwon used Kernel ridge regression (KRR) to learn the mapping from the low-resolution image to target the high-resolution image based on the example pairs of input and output images [68]. With the recent advancement of machine learning, various Convolutional Neural Network (CNN) based algorithms have been proposed [69–71].

Multi-frame SR uses many low-resolution (LR) images to create one high-resolution(HR) image. The main idea in multi-frame SR is to use the complementary information between multiple images with sub-pixel misalignment to construct an HR image of a scene. These global subpixel displacements happen due to the relative positions of the cameras and also due to camera motion, such as panning, zooming or minimal rotation. Multi-frame SR has the advantage over single-frame SR as more information is available, and no learning is needed to construct the target HR image. Fig 5.1 conveys this idea by super-resolving the three 3x3 pixel images to form one 4x4 pixel image.

Multi-frame SR is usually an ill-posed inverse problem, and researchers have developed several algorithms to solve this problem. Traditionally, SR was done in frequency domain [59, 72, 73] and typically relied on familiar Fourier transform properties. It is easy to grasp and very appealing, and its ability to be implemented in parallel means reduced hardware complexity. However, this method is sensitive to model errors, difficult to handle complicated motion models, and restricted to only translational motion [74]. It is also hard

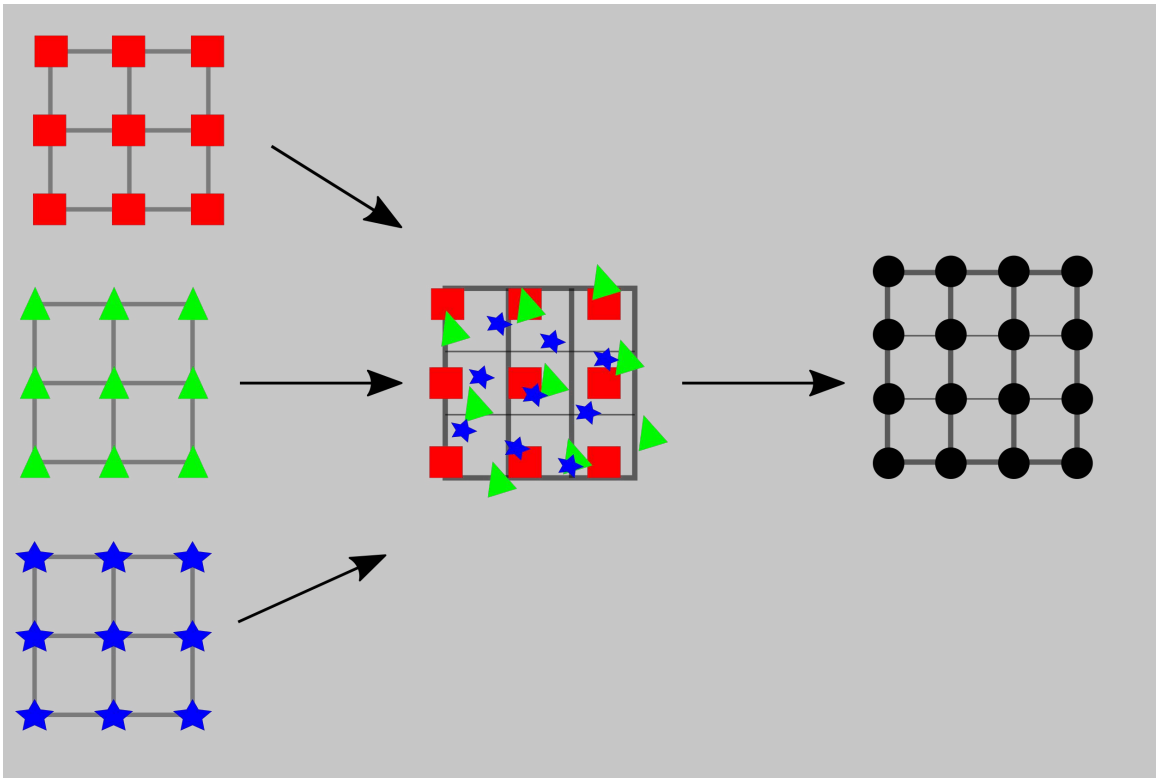


Fig. 5.1: illustration of superresolution.

to include spatial domain *a priori* knowledge for regularization. Due to these drawbacks, researchers concentrate more on spatial domain SR.

Spatial domain SR formulates the observation model of the imaging system, and reconstruction of the HR image is done in the spatial domain. There are several ways to do SR in the spatial domain, including Iterative Back Projection (IBP) [75], Projection Onto Convex set (PCOS) [76–78], regularized methods [79–81] and interpolation-restoration methods [82]. In IBP, the HR image is estimated iteratively by backprojecting the difference between observed and simulated LR images. The simulated LR images are formed using the estimated HR image, and the process is carried out iteratively until the defined error function is minimum. IBP is intuitively easy to understand and implement. However, the solution of IBP has no unique solution and is harder to include *a priori* constraints. It is also difficult to choose a proper back-projection kernel, as the choice of this Kernel affects the characteristics of the solution. Researchers proposed an alternative to IBP called PCOS. In this method,

each constraint based on each prior is defined as a non-empty convex set, and the solution lies in the intersection of these convex sets. The solution is found iteratively by projecting an initial estimate and successive estimates onto all of these convex sets. Even though it is easy to incorporate all the priors, the slow convergence and the lack of a unique final solution poses a major disadvantage. The regularized method is more popular among researchers as the inclusion of regularization term usually makes the SR problem well-posed. An objective function is formed using the data fidelity and regularization terms. A Lagrange multiplier, also called a regularization parameter, balances between those terms and a unique solution is found by minimizing the objective function. According to the Bayesian theorem, the regularization term represents the image prior modeling, providing the prior knowledge about the desired image. The regularization term plays a vital role in solving the problem of SR, and therefore, many different prior models have been proposed [83–85].

Interpolation-based methods are the most intuitive and computationally efficient in calculating each HR pixel. This method consists of three steps: registration, interpolation and restoration. During registration, the relative motions between each LR frame are estimated. This step helps transform all the LR images into a common non-uniform grid to interpolate HR pixels. The reconstruction procedure can then be applied to produce uniformly spaced HR pixels. Finally, the restoration step is carried out to deblur the image and to remove the noise [86]. The restoration process is important as it restores the high-frequency detail lost from the image.

This work focused on SR on each triangle instead of the whole image. The triangle-based texturing showed a promising result as seen in Chapter 4. The use of SR on triangles is particularly effective because it allows for different upscale factors based on the number of triangles available for SR. With more triangle textures available for a given 3D triangle, more samples of the LR pixel can contribute to sub-pixel displacement. For example, when dealing with terrains, a triangle on a flat area will be seen by more images, and thus will have more LR pixels available than a triangle on a cliff. Using the same upscale factor for both cases is not optimal, as it will not yield the best results. Therefore, by utilizing

different scale factors, the quality of the output can be improved.

5.1 Adaptive Wiener Filter

Since the proposed work exploits a sequence of images, multi-image SR is vital to produce accurate and better results, as the final image will have the resolution at the sub-pixel level instead of the pixel level. Since the UAS is flying much higher than the ground and takes images rapidly, there is no significant texture change from one image to the next. This makes superresolution possible as there will be sub-pixel motion between swaths. There will be many pixels with sub-pixel displacement comprising the same object. This section discusses using interpolation-based superresolution to upscale the triangle texture used to texture each 3D mesh triangle. In particular, we focused on using the Adaptive Wiener Filter (AWF), which is a minimum mean square filter, and applying its principle to determine the value of each HR pixel in the final texture [87]. The term adaptive is used because the filter weights are different for each HR pixel, and it changes based on the value of LR pixels and the spatial distribution of LR pixels that comprise each HR pixel. The results from the registration step described in Chapter 3 are used as the registration step of superresolution.

Most interpolation-based SR techniques treat registration, interpolation, and restoration separately. However, the Wiener filter technique combines interpolation and restoration into a weighted sum operation. This algorithm uses sub-pixel registration to register all the LR images onto an HR grid. A moving window is employed, and a Wiener filter is applied to each window. HR pixels are computed using a weighted sum of LR pixels based on the weights computed within this window. The statistics for the Wiener filter change every window as it is based on the spatial position of LR pixels and the variance of the pixel values within that window. The basic overview of the algorithm is shown in Fig 5.2

Before we describe the Wiener filter for SR, we need to understand the relationship between the target HR image and the distorted LR images obtained from the HR image.

5.2 Methodology

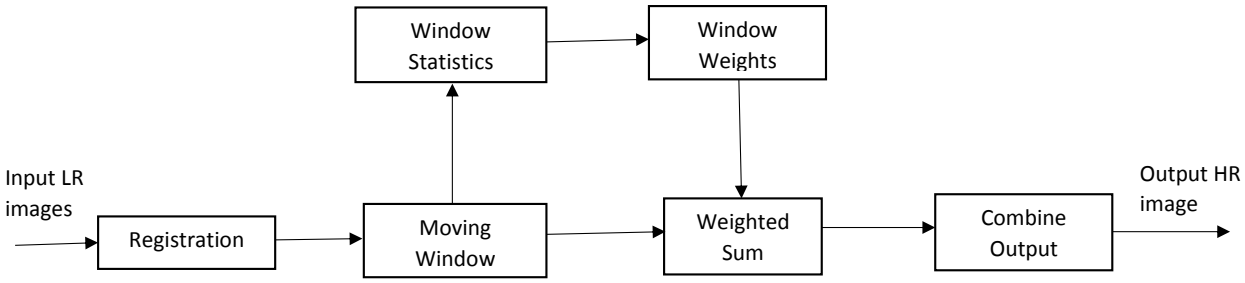


Fig. 5.2: Basic algorithm overview of SR using Wiener filter

The SR algorithm described in this section is also an interpolation method, but it is performed in each triangle with a varying upscale factor. In the texel camera domain, previous works did not take advantage of the availability of multiple images that describe the same scene in the ground. This chapter presents an SR algorithm with the imaging system modeled by a simple point spread function (PSF). This simple model helps to demonstrate the concept of triangle-based SR. To create a TDSM, registration of multiple texel images is required. Since registration is the first step in SR using an interpolation-based approach, the registration parameters obtained from registering the texel images can be reused to reduce computation. Fig 5.3 shows the overall algorithm for creating a TDSM with super-resolved image.

The first step to SR is to register all texel images properly. For triangle-based SR, the 3D triangle that needs to be textured is back-projected onto all the images where it is visible. From Chapter 4, we can determine all the 2D coordinates of the three vertices associated with the 3D mesh triangle. Since SR heavily depends on the registration quality, finding the correct pose of each texel image is crucial in the pre-superresolution stage.

The minimum mean square filter, commonly known as the Wiener filter described in this section, is used to estimate the HR pixel based on all the available LR pixels. First, a triangle in the HR grid is created. This triangle is the scaled version of the best triangle

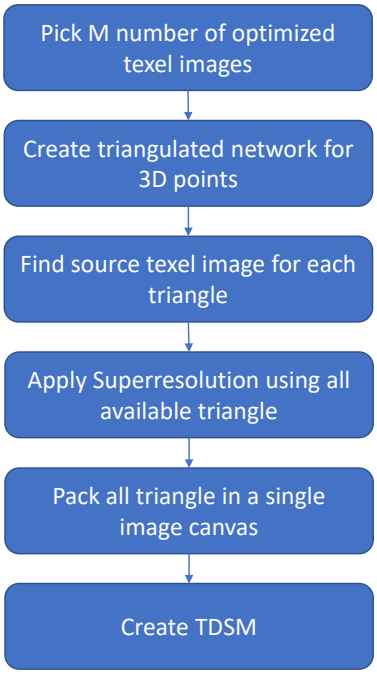


Fig. 5.3: Flowchart showing the creation of a TDSM using the optimized texel images.

that was found in Chapter 4. The scale factor, Sn, is a user-defined parameter dependent on the number of triangle textures associated with the 3D mesh triangle. Next, all the pixels from each LR triangle are transformed to the target triangle in the HR grid. The transformation between the LR triangle and the HR triangle is found using three vertices of each triangle. Since those three vertices correspond to the same locations in the 3D mesh triangle, an affine transformation can be found using three pairs of corresponding vertices. This transformation is then used to transform all the pixels within the LR triangle to the HR grid.

In the HR triangle, a small sliding window with size $W_x \times W_y$ called the observation window is employed. All the LR pixels that lie within this observation window are placed into a vector $\mathbf{g}_i = [g_{1_i}, g_{2_i}, \ldots, g_{k_i}]^T$, where k is the number of LR pixels within the i^{th} window. We wish to estimate the HR pixel that lies at the center of this observation window from the pixels in $\mathbf{g_i}$. If $\hat{d}_i$ is the estimated HR pixel, then it can be written as a

linear combination of the LR pixels in $\mathbf{g}_i$ as

$$\hat{d}_i = c_1 g_{1_i} + c_2 g_{2_i} + \ldots + c_k g_{k_i}. \tag{5.1}$$

If d_i is the true HR pixel, then we can write:

$$\begin{aligned} d_i &= \hat{d}_i + e \\ &= c_1 g_1 + c_2 g_2 + \ldots + c_k g_k + e \\ &= \sum_{j=0}^{k} c_j g_j + e, \end{aligned} \tag{5.2}$$

where e is the approximate error such that the norm of the error, $\|e\|$, is as small as possible. The orthogonality principle states that the norm of the error is minimized when the error is orthogonal to the data that forms our estimate of the HR pixel. Mathematically, $\|e\|$ is minimized when,

$$e \perp (g_1, g_2, \ldots, g_k)$$

i.e.

$$\begin{aligned} \langle d_i - \sum_{j=0}^{k} c_j g_j, g_1 \rangle &= 0 \\ \langle d_i - \sum_{j=0}^{k} c_j g_j, g_2 \rangle &= 0 \\ &\vdots \\ \langle d_i - \sum_{j=0}^{k} c_j g_j, g_k \rangle &= 0 \end{aligned} \tag{5.3}$$

where $\langle ., . \rangle$ is the inner product.

After expanding and simplifying (5.3), we can write

$$\begin{aligned} c_1\langle g_1, g_1\rangle + c_2\langle g_2, g_1\rangle + \ldots + c_k\langle g_k, g_1\rangle &= \langle d_i, g_1\rangle \\ c_1\langle g_1, g_2\rangle + c_2\langle g_2, g_2\rangle + \ldots + c_k\langle g_k, g_2\rangle &= \langle d_i, g_2\rangle \\ &\vdots \\ c_1\langle g_1, g_k\rangle + c_2\langle g_2, g_k\rangle + \ldots + c_k\langle g_k, g_k\rangle &= \langle d_i, g_k\rangle. \end{aligned} \tag{5.4}$$

Writing in matrix form, we get

$$\begin{bmatrix} \langle g_1, g_1\rangle & \langle g_2, g_1\rangle & \ldots & \langle g_k, g_1\rangle \\ \langle g_1, g_2\rangle & \langle g_2, g_2\rangle & \ldots & \langle g_k, g_2\rangle \\ \vdots & & & \vdots \\ \langle g_1, g_k\rangle & \langle g_2, g_k\rangle & \ldots & \langle g_k, g_k\rangle \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} \langle d_i, g_1\rangle \\ \langle d_i, g_2\rangle \\ \vdots \\ \langle d_i, g_k\rangle \end{bmatrix}. \tag{5.5}$$

If we assume $\mathbf{g}$ and d_i as random variables, then we can define the above relationships in a probabilistic framework. Let us define the inner product in terms of Expectation as

$$\langle X, Y\rangle = \mathbb{E}[XY^T]. \tag{5.6}$$

In this case, the square norm of the error, $\|e\|^2$, is termed as mean square error and is given as $\mathbb{E}[ee^T]$. Using the inner product defined in (5.6), (5.5) can be written as

$$\begin{bmatrix} \mathbb{E}[g_1 g_1^T] & \mathbb{E}[g_2 g_1^T] & \ldots & \mathbb{E}[g_k g_1^T] \\ \mathbb{E}[g_1 g_2^T] & \mathbb{E}[g_2 g_2^T] & \ldots & \mathbb{E}[g_k g_2^T] \\ \vdots & & & \vdots \\ \mathbb{E}[g_1 g_k^T] & \mathbb{E}[g_2 g_k^T] & \ldots & \mathbb{E}[g_k g_k^T] \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} \mathbb{E}[d_i g_1^T] \\ \mathbb{E}[d_i g_2^T] \\ \vdots \\ \mathbb{E}[d_i g_k^T] \end{bmatrix}$$

$$R\mathbf{c} = \mathbf{p} \tag{5.7}$$

$$\mathbf{c} = R^{-1}\mathbf{p}, \tag{5.8}$$

where $\mathbf{R} = \mathbb{E}\{\mathbf{g}_i\mathbf{g}_i^T\}$ is the auto-correlation matrix of the vector of LR pixels inside i^{th} observation window, $\mathbf{p} = \mathbb{E}\{\mathbf{g}_i d_i^T\}$ is the vector of cross-correlation between desired HR pixel and input LR pixels and $\mathbf{c}$ is the vector of coefficients. So the problem of finding the value of the HR pixel in the center of the i^{th} observation window such that the mean square error is minimum is reduced to finding the elements of $\mathbf{R}$ and $\mathbf{p}$ and solving (5.7).

5.3 Spatially-Varying Statistical model

To compute the coefficient for (5.7), we need to compute the required statistics to fill $\mathbf{R}$ and $\mathbf{p}$. One way to do this is to use the training images whose statistics are assumed to be the same as the real data set we are interested in. The autocorrelation and cross-correlation are computed empirically using these training sets of images [88]. The required statistics can be populated for the full observation window at a certain resolution. The elements of $\mathbf{R}$ and $\mathbf{p}$ for each observation window can then be sub-sampled from this fully populated autocorrelation matrix and cross-correlation vector. This method works well if the training set has the same or very similar statistics as the data set of interest. However, the coefficients will be poorly computed if the training set is significantly different than the data set of interest.

Another method is to employ a continuous parametric model for the autocorrelation function for the desired HR image. All the required statistics are derived from this model to fill in the elements of $\mathbf{R}$ and $\mathbf{p}$ based on the location of LR pixels on the HR grid. This method eliminates the need for a training set and has a small number of tuning parameters.

Let $\mathbf{f}_i = [f_{1_i}, f_{2_i}, \ldots, f_{k_i}]^T$ be the noise free version of $\mathbf{g}_i$ and let $\mathbf{n}_i$ be the random noise vector associated with i^{th} observation window and uncorrelated with $\mathbf{f}_i$. We assumed this noise to be independent and identically distributed (iid) with zero mean and a variance of σ_n^2. We have

$$\mathbf{g}_i = \mathbf{f}_i + \mathbf{n}_i. \tag{5.9}$$

Thus,

$$\begin{aligned}\mathbf{R} &= \mathbb{E}\{\mathbf{g}_i\mathbf{g}_i^T\} \\ &= \mathbb{E}\{(\mathbf{f}_i+\mathbf{n}_i)(\mathbf{f}_i+\mathbf{n}_i)^T\} \\ &= \mathbb{E}\{\mathbf{f}_i\mathbf{f}_i^T\} + \sigma_n^2\mathbf{I} \\ &= \begin{bmatrix} \mathbb{E}[f_1f_1^T] & \mathbb{E}[f_2f_1^T] & \dots & \mathbb{E}[f_kf_1^T] \\ \mathbb{E}[f_1f_2^T] & \mathbb{E}[f_2f_2^T] & \dots & \mathbb{E}[f_kf_2^T] \\ \vdots & & & \vdots \\ \mathbb{E}[f_1f_k^T] & \mathbb{E}[f_2f_k^T] & \dots & \mathbb{E}[f_kf_k^T] \end{bmatrix} + \sigma_n^2\mathbf{I}\end{aligned} \tag{5.10}$$

and

$$\begin{aligned}\mathbf{p} &= \mathbb{E}\{\mathbf{g}_i d_i^T\} \\ &= \mathbb{E}\{(\mathbf{f}_i+\mathbf{n}_i)d_i^T\} \\ &= \mathbb{E}\{\mathbf{f}_i d_i^T\} \\ &= \begin{bmatrix} \mathbb{E}[d_if_1^T] \\ \mathbb{E}[d_if_2^T] \\ \vdots \\ \mathbb{E}[d_if_k^T] \end{bmatrix}.\end{aligned} \tag{5.11}$$

In theory, $\mathbf{f_i}$ are the samples of $f(x,y)$, a blurred version of the original HR image, $d(x,y)$. This blurring can be caused by diffraction from the optics and other factors like motion blur, defocus and even atmospheric conditions. In this work, we only consider a simple PSF that models the diffraction caused by the optics as shown in Fig 5.4. If $h(x,y)$ is the PSF of the imaging system, then we can write

$$f(x,y) = d(x,y) * h(x,y). \tag{5.12}$$

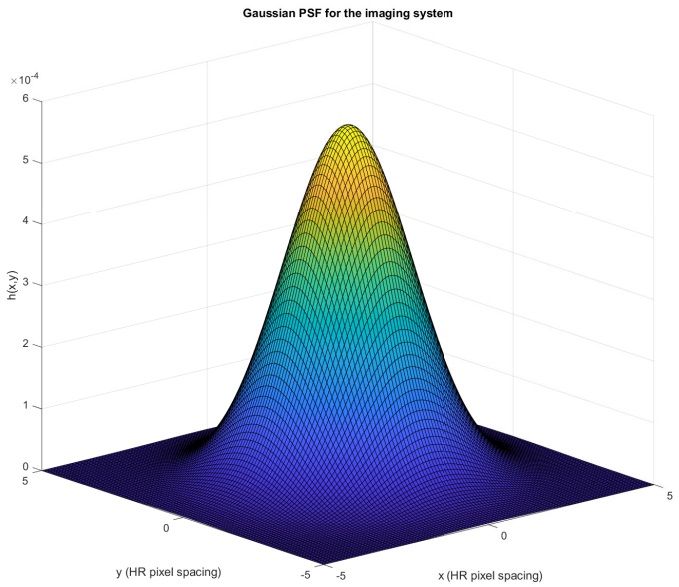


Fig. 5.4: Gaussian PSF of the imaging system, $h(x, y)$ with $\mu_x = \mu_y = 0$ and $\sigma_x = \sigma_y = 1.5$ pixels

Let us consider a wide sense stationary (WSS) signal, $r_{dd}(x, y)$, that models the auto-correlation function for the desired HR image,$d(x, y)$. The cross-correlation between $d(x, y)$ and $f(x, y)$ can be given as

$$r_{df}(x, y) = \mathbb{E}[d(x, y)f(x, y)]. \tag{5.13}$$

Similarly, autocorrelation for $f(x, y)$ is given as

$$r_{ff}(x, y) = \mathbb{E}[f(x, y)f(x, y)]. \tag{5.14}$$

Using (5.12) and simplifying, we can express $r_{df}(x,y)$ and $r_{ff}(x,y)$ in terms of $r_{dd}(x,y)$ as

$$r_{df}(x,y) = r_{dd}(x,y) * h(x,y) \tag{5.15}$$

$$r_{ff}(x,y) = r_{dd}(x,y) * h(x,y) * h(-x,-y). \tag{5.16}$$

In this work, we employ a circularly symmetric function to model the autocorrelation for the desired image, which is of the form

$$r_{dd}(x,y) = \sigma_{d_i}^2 \rho^{\sqrt{x^2+y^2}}, \tag{5.17}$$

where $\sigma_{d_i}^2$ is the desired variance for the HR pixel in the i_{th} observation window, and ρ is the tuning parameter that corresponds to one pixel step correlation value in the desired HR image. This parameter controls the decay of the autocorrelation value with the spatial distance. Figure 5.5 shows the autocorrelation model employed in this study. By substituting the value of $r_{dd}(x,y)$ and $h(x,y)$ in (5.16), we can fill the elements of $\mathbf{R}$ based on the vertical and horizontal distance between all LR pixels within g_i. Thus, $\mathbf{R}$ can be computed as

$$\mathbf{R} = \begin{bmatrix} r_{ff}(0,0) & r_{ff}(\triangle(2,1)) & \dots & r_{ff}(\triangle(k,1)) \\ r_{ff}(\triangle(1,2)) & r_{ff}(0,0) & \dots & r_{ff}(\triangle(k,2)) \\ \vdots & & & \vdots \\ r_{ff}(\triangle(1,k)) & r_{ff}(\triangle(2,k)) & \dots & r_{ff}(0,0) \end{bmatrix} + \sigma_n^2 \mathbf{I}, \tag{5.18}$$

where $\triangle(m,n) = [\triangle_x(g_m,g_n), \triangle_y(g_m,n)]$ and $\triangle_x(g_m,g_n)$ and $\triangle_y(g_m,g_n)$ are the x and y distance in unit of pixels between LR pixels g_m and g_n respectively.

Similarly, by substituting the value of $r_{dd}(x,y)$ and $h(x,y)$ in (5.15), we can fill the elements of $\mathbf{p}$ based on the vertical and horizontal distance between each LR pixels and HR

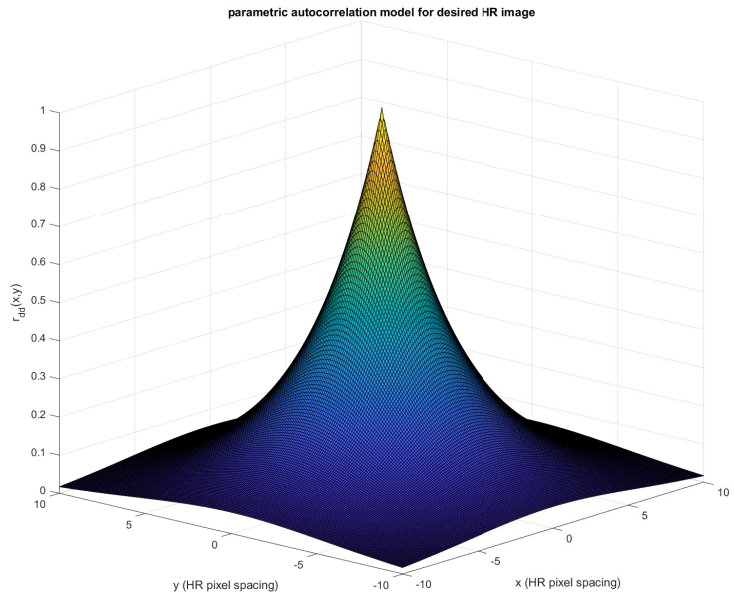


Fig. 5.5: Autocorrelation model for desired HR image, $r_{dd}(x, y)$ with $\sigma_d^2 = 1$ and $\rho = 0.75$

pixel in the middle of the observation window. Thus, $\mathbf{p}$ can be computed as

$$\mathbf{p} = \begin{bmatrix} r_{df}(\triangle(1, d_i)) \\ r_{df}(\triangle(2, d_i)) \\ \vdots \\ r_{df}(\triangle(k, d_i)) \end{bmatrix}, \tag{5.19}$$

where $\triangle(m, d_i) = [\triangle_x(g_m, d_i), \triangle_y(g_m, d_i)]$ and $\triangle_x(g_m, d_i)$ and $\triangle_y(g_m, d_i)$ are the x and y distance in unit of pixels between LR pixel g_m and HR pixel d_i that is situated in the middle of i_{th} observation window.

The final information to compute the weight of the filter is to estimate the value of $\sigma_{d_i}^2$. Instead of choosing a global value of σ_d^2, we estimate it based on the value of the LR pixel within the observation window. This way, the weights adapt not only to the spatial distribution of LR pixels but also to the intensity of the LR pixels within the observation window. If $\hat{\sigma}_{g_i}^2$ is the sample variance of LR pixels in $\mathbf{g}_i$ and $\sigma_{n_i}^2$ is the noise variance, then

the variance for noise-free LR pixels can be estimated as

$$\hat{\sigma}_{f_i}^2 = \hat{\sigma}_{g_i}^2 - \sigma_n^2. \tag{5.20}$$

Using (5.16), we can estimate the desired image variance, $\hat{\sigma}_{d_i}^2$, in terms of $\hat{\sigma}_{f_i}^2$ as

$$\hat{\sigma}_{d_i}^2 = \frac{1}{C(\rho)} \hat{\sigma}_{f_i}^2, \tag{5.21}$$

where $C(\rho)$ is the normalizing factor and is given as,

$$C(\rho) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \rho^{\sqrt{x^2+y^2}} \left\{ h(x,y) * h(-x,-y) \right\} dxdy. \tag{5.22}$$

Now we have all the pieces required to calculate $\mathbf{R}$ and $\mathbf{p}$, which can be substituted into (5.8) to calculate the weight of the filter. The weights are normalized such that the sum of all weights is equal to 1 to remove any potential grid artifact. Finally, these weights are substituted into (5.1) to calculate the value of the HR pixel such that the mean square error is minimum.

This process is repeated to estimate every HR pixel in the triangle. The spatial distribution of LR pixels within the observation window is changed in each iteration which changes the weights accordingly.

5.4 Results and Discussion

In this section, we demonstrate the use of the Wiener filtering approach by improving the resolution of an image. The results are shown for a small area of the final TDSM which consists of 21 triangles. The area is chosen such that it consists of a flat area as well as edges. As described in Section 5.2, we first use the registration parameter found in Chapter 3 to find all the textures associated with any 3D mesh triangle. Each 3D mesh triangle is back-projected to all the images using the optimized 3D vertex points and the optimized pose for each camera. The LR pixels from the images where this 3D mesh triangle is visible are used in the superresolution algorithm. An example of a 3D mesh triangle is shown in Fig 5.6.

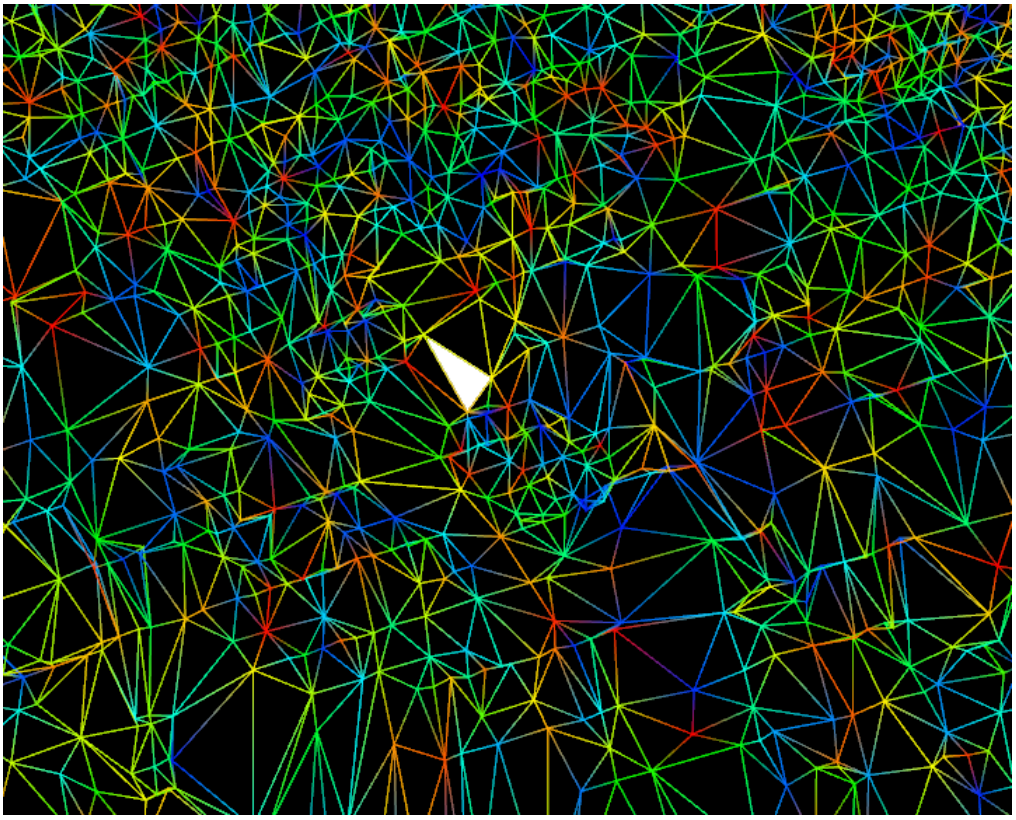


Fig. 5.6: 3D mesh triangles

The 3D mesh triangle with a solid white face is back-projected to all the images. Out of 21 images used to create a TDSM, it is visible in 11 images, as shown in Fig 5.7. The images are ordered based on the scoring metric for each image discussed in Chapter 4. Fig 5.8 shows more details after it is zoomed. The texture inside the red triangle corresponds to the 3D mesh triangle, and the super-resolved texture is created using the pixels from these LR triangles.

The LR triangle with the highest score is chosen as the shape for the final HR triangle. The superresolution factor, S_f, is chosen to be two, and the best LR triangle is scaled by a factor of 2. All the pixels from the best LR triangle, as shown in Fig 5.9, are then transformed to the HR triangle in the HR grid. Figure 5.9 is zoomed in to show all individual pixels in Fig 5.10, which will then be transformed to the HR triangle. Since the transformation is just the scaling, these pixels lie in a regular grid in the HR triangle, as shown in Fig 5.11.

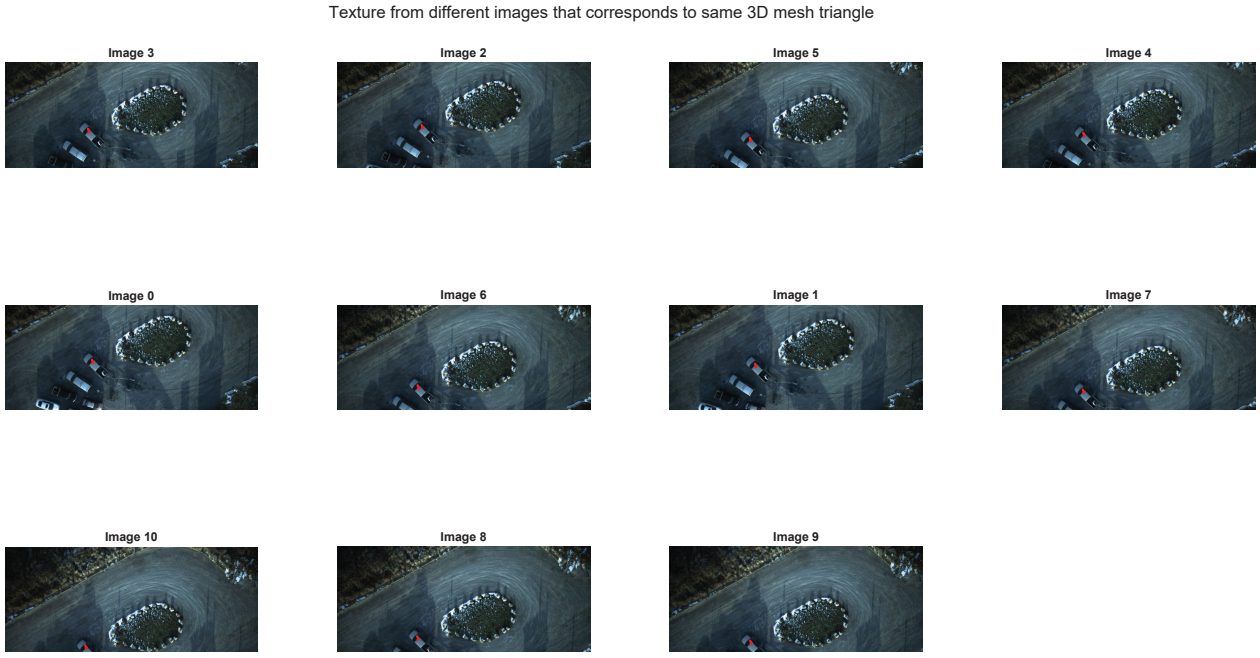


Fig. 5.7: 3D triangle projected onto all images sorted using score

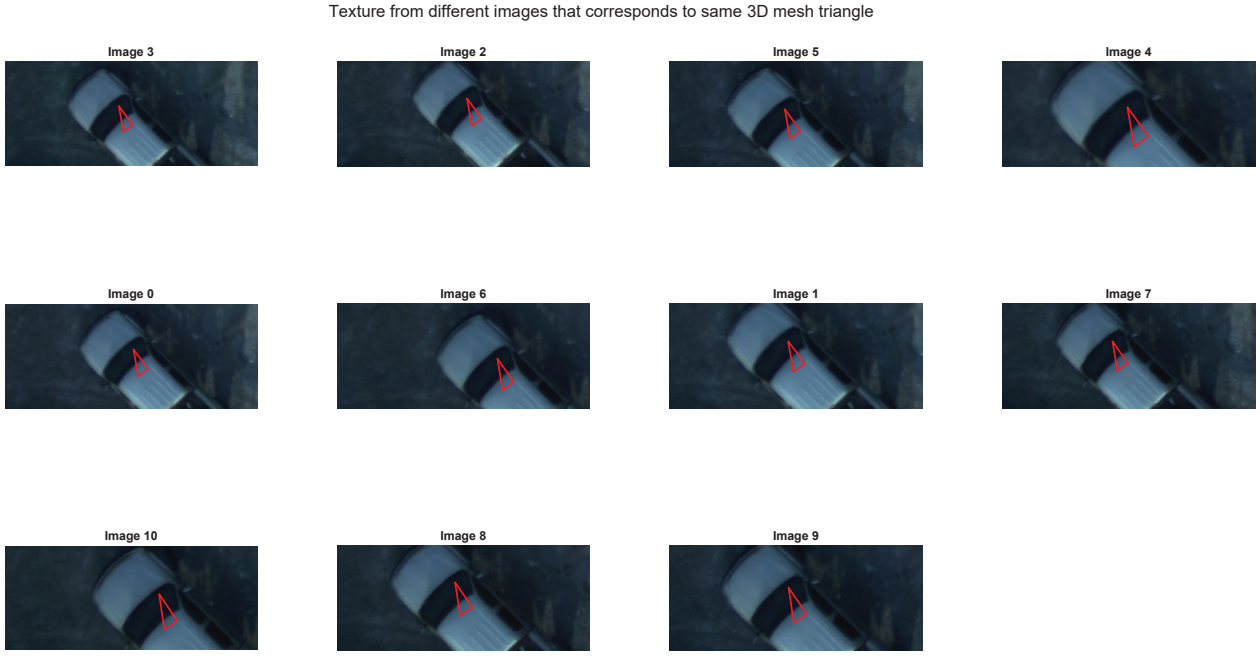


Fig. 5.8: Zoomed to show details

To transform all the pixels from the remaining LR triangle, we need to find the transformation between each LR triangle and the HR triangle. Since all the LR triangles are the projection of the same 3D mesh triangle, the three vertices of each LR triangle and the HR triangle correspond to the same three vertices in a 3D mesh triangle. An affine transformation can be found using three corresponding points. All remaining pixels inside the LR triangle can then be transformed to the HR triangle using this transformation. Figure 5.12 shows the HR triangle in the HR grid where small dots inside the triangle correspond to each LR pixel from all remaining LR triangles.

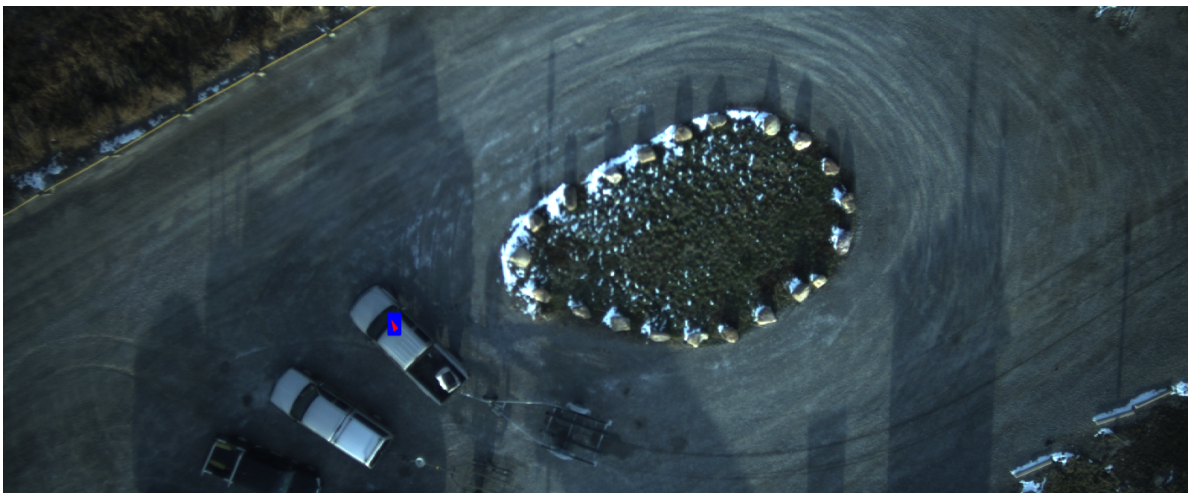


Fig. 5.9: Best LR triangle in original LR image. Red pixel corresponds to all the pixel inside the triangle and blue corresponds outside the triangle within the bounding box.

After all the LR pixels are transformed to the HR grid, a sliding observation window of $3x3$ pixels is employed from the top left to the bottom right of the HR triangle. An example of such an observation window denoted by a red box is shown in Fig 5.13. In each iteration, a Wiener filter described in section is applied to the observation window, and the value of the HR pixel in the middle of the observation window is computed. The value of the HR pixel is dependent only on the pixels inside the observation window. This process is repeated to estimate the value of all the HR pixels inside the triangle.

Figure 5.14 shows one instance of an observation window. The stars are the LR pixels

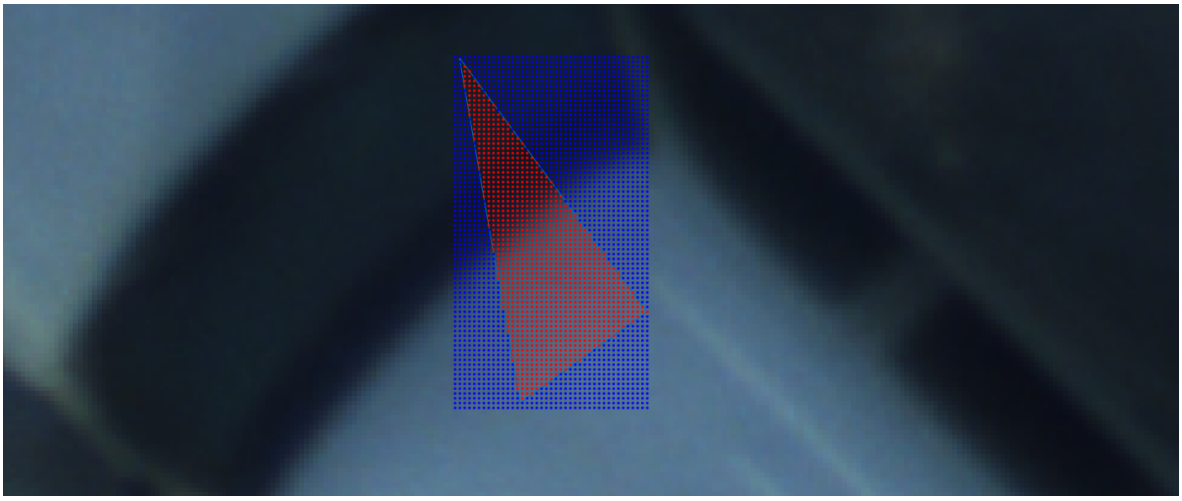


Fig. 5.10: Fig 5.9 is zoomed to show details

inside this window, and the red rectangle is the HR pixel we are trying to estimate. Each color corresponds to a pixel from different LR images. Figure 5.15 and 5.16 show the distribution of the LR pixels in the observation window and the output weights associated with each LR pixel. The weights adapt based on the spatial distribution of the LR pixels and the value of each LR pixel inside this window.

Figure 5.17 shows the result of the algorithm applied to one triangle. The original best LR triangle is shown in Fig 5.17(a), and the super-resolved HR triangle is shown in Fig 5.17(b). We can see the HR triangle is visually better than the LR triangle. The output triangle obtained with single frame bilinear interpolation is shown in Fig 5.17(c). In this case, best triangle is upscaled using interpolation by a factor of two. Figures 5.18, 5.19, 5.20, and 5.21 show more results after applying SR algorithm. In each case S_f is chosen to be two. The HR triangle is significantly better than the LR triangle in each of these results. Finally, Fig 5.22 shows the final TDSM result after using all the triangle textures. Fig 5.22(a) shows the wireframe of the TDSM, which was constructed by triangulating the lidar point cloud. These 3D triangles were filled using the LR texture triangle shown in Fig 5.22(b) and with the HR triangle texture shown in Fig 5.22(c). The 3D viewer program automatically apply interpolation to the texture as we zoom in on the TDSM created using LR triangles.

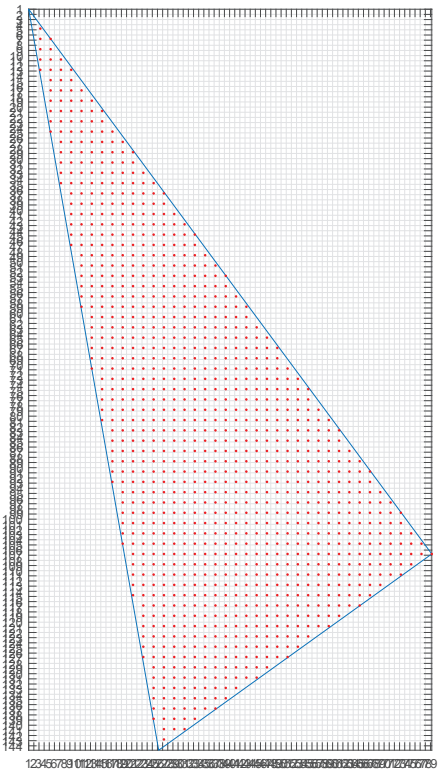


Fig. 5.11: LR pixels from best triangle transformed to HR grid

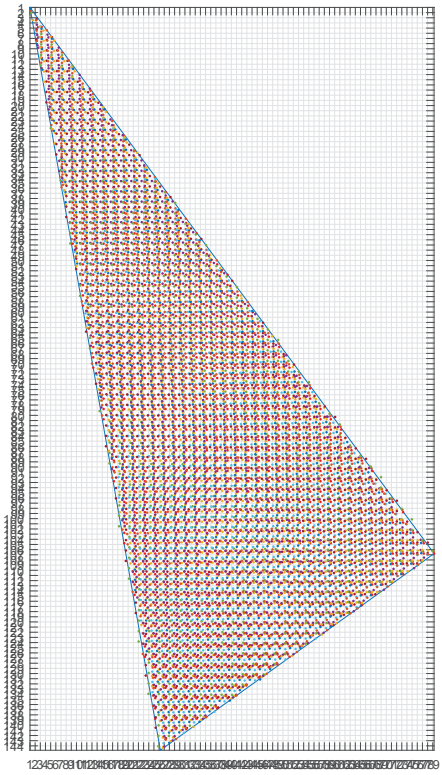


Fig. 5.12: LR pixels from all triangles transformed to HR grid

Fig. 5.13: Zoomed to show details. The red square comprises of 3 x 3 HR pixels is an instance of an observation window

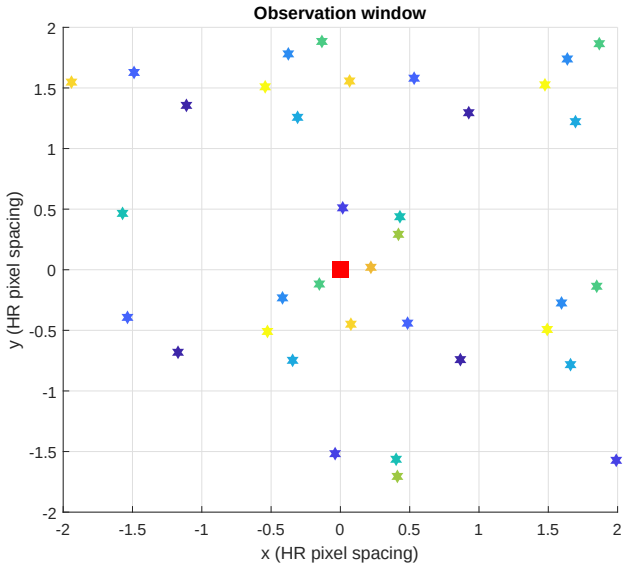


Fig. 5.14: Observation window. Each color corresponds to a pixel from a different LR image

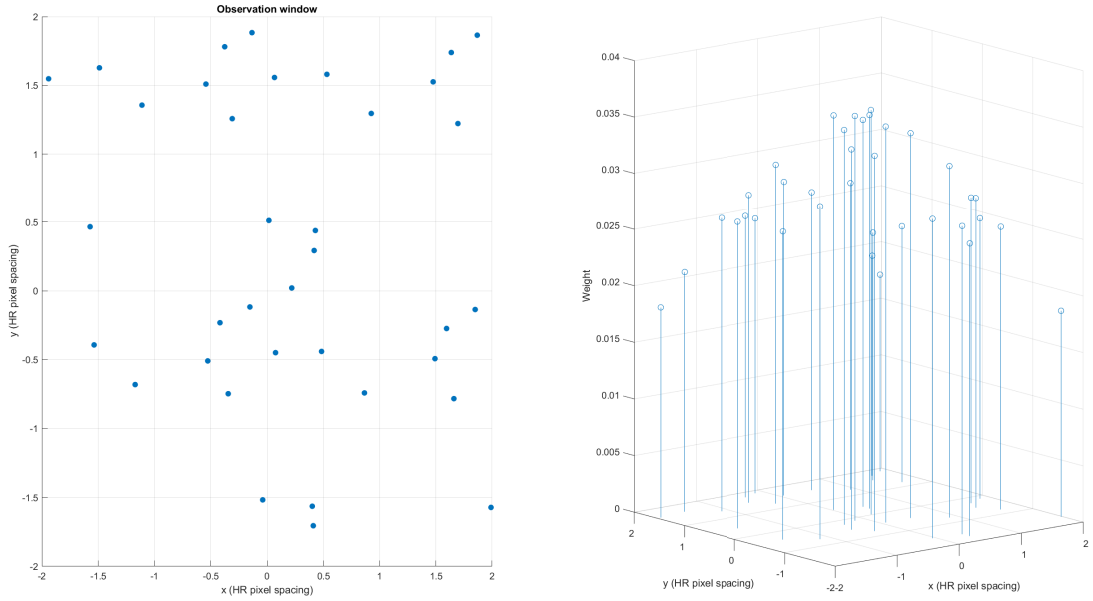


Fig. 5.15: Weights based on the observation window

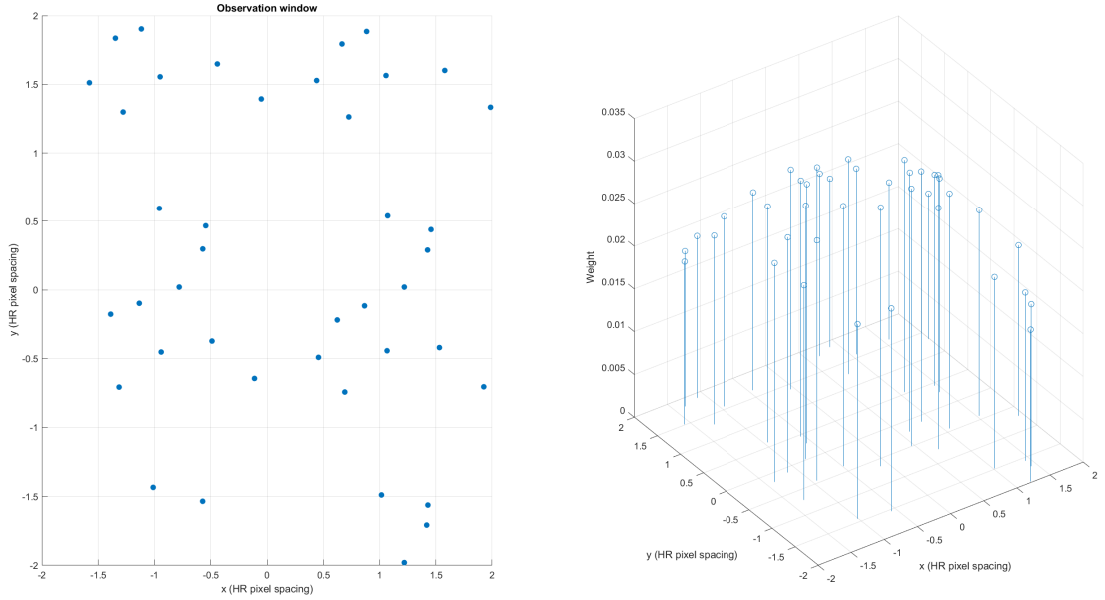
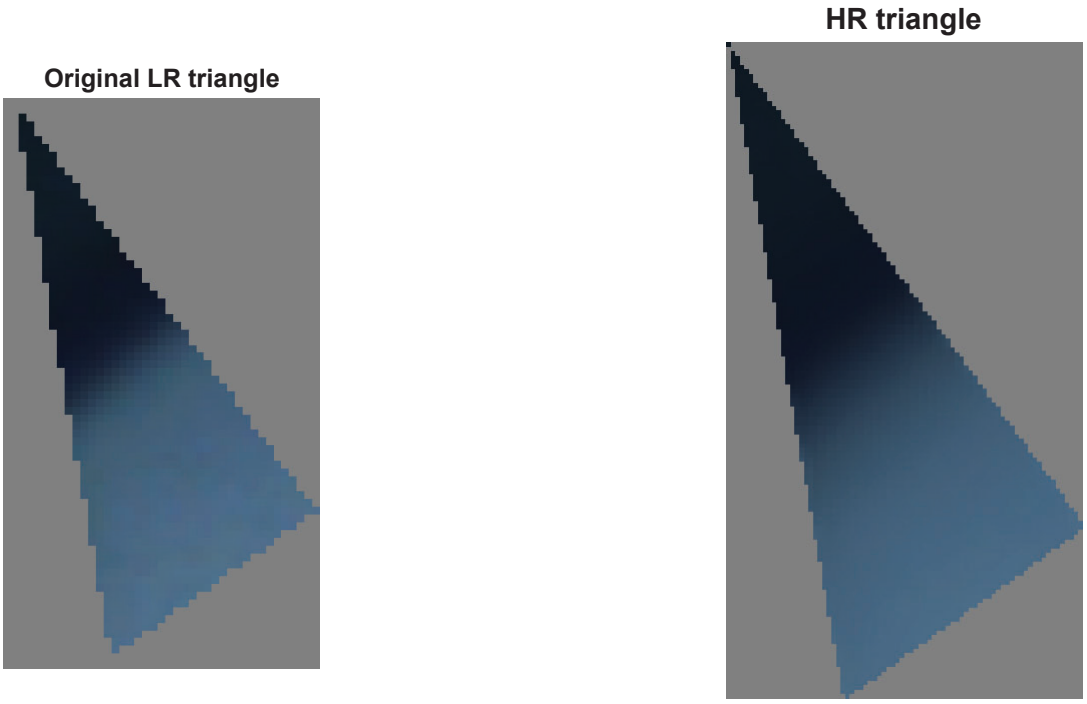
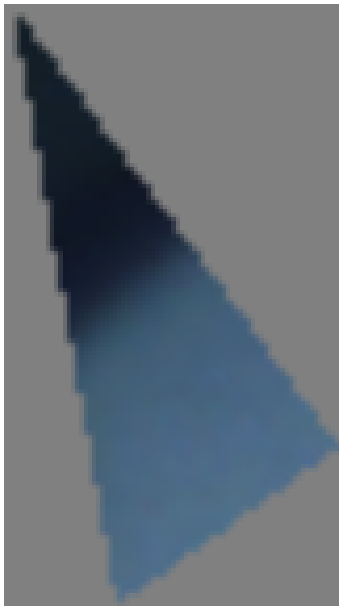


Fig. 5.16: Weights based on the observation window

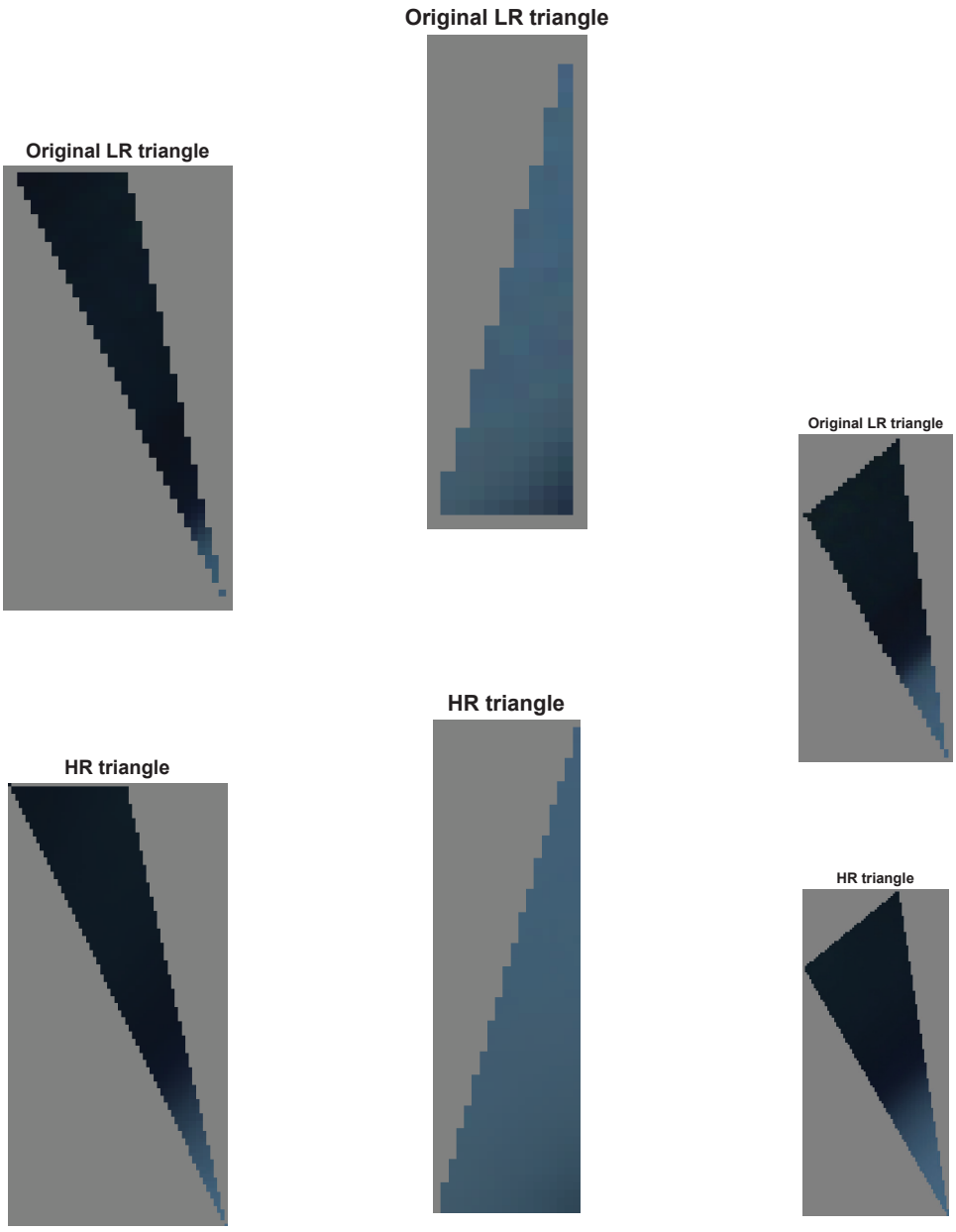


(a) (b)



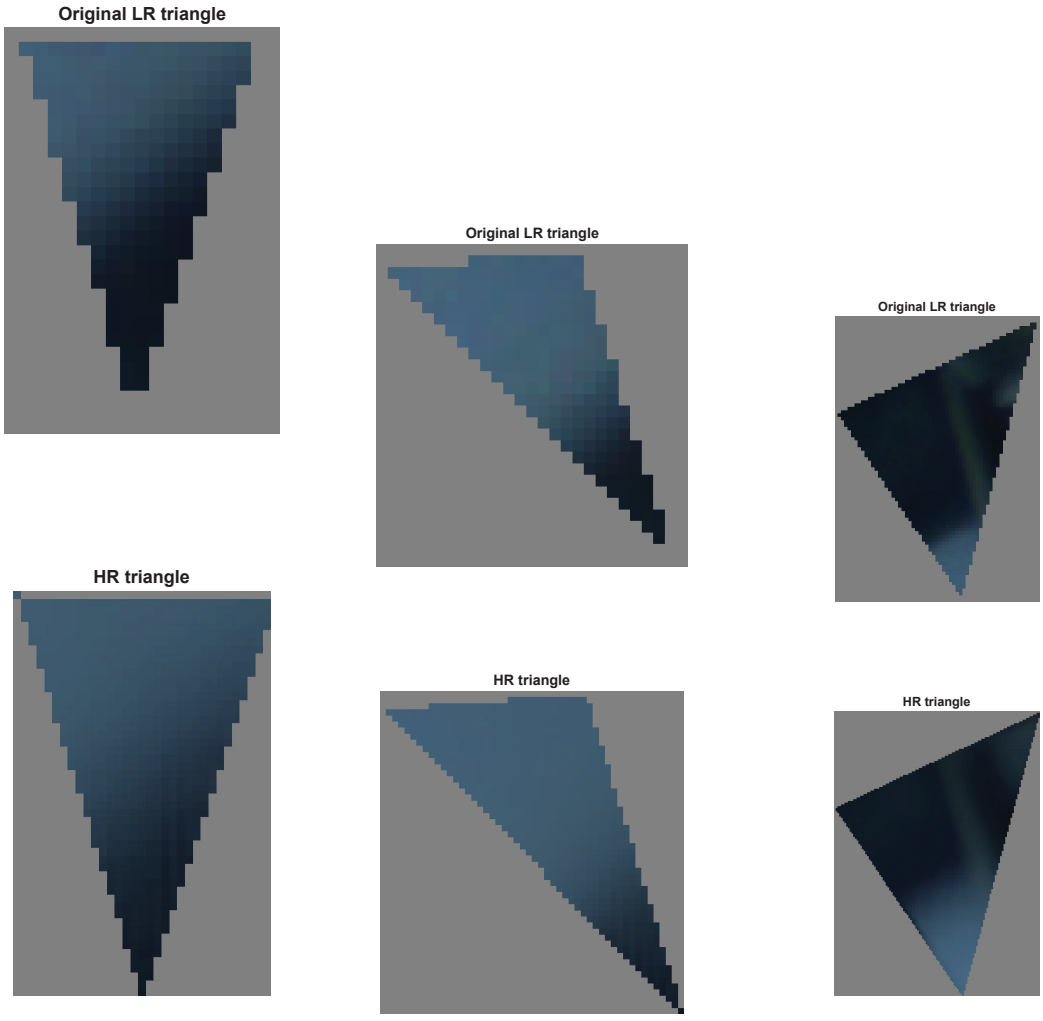
(c)

Fig. 5.17: (a) Original LR triangle (b) HR triangle after applying the algorithm (c) Bilinear interpolated triangle applied to LR triangle



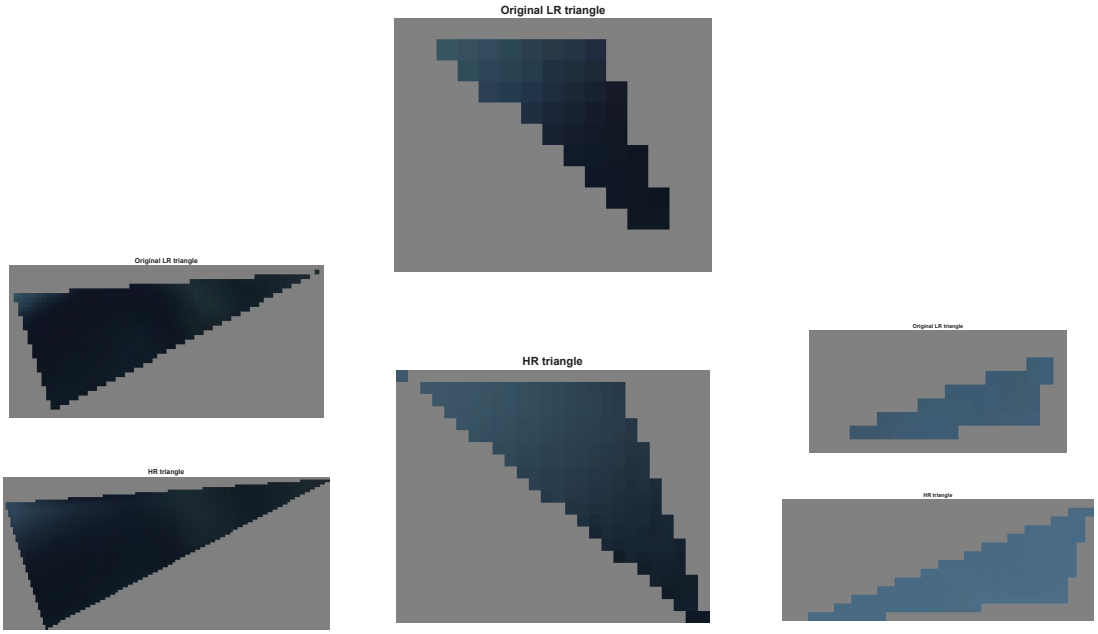
(a) image1 (b) image2 (c) image3

Fig. 5.18: Results 1



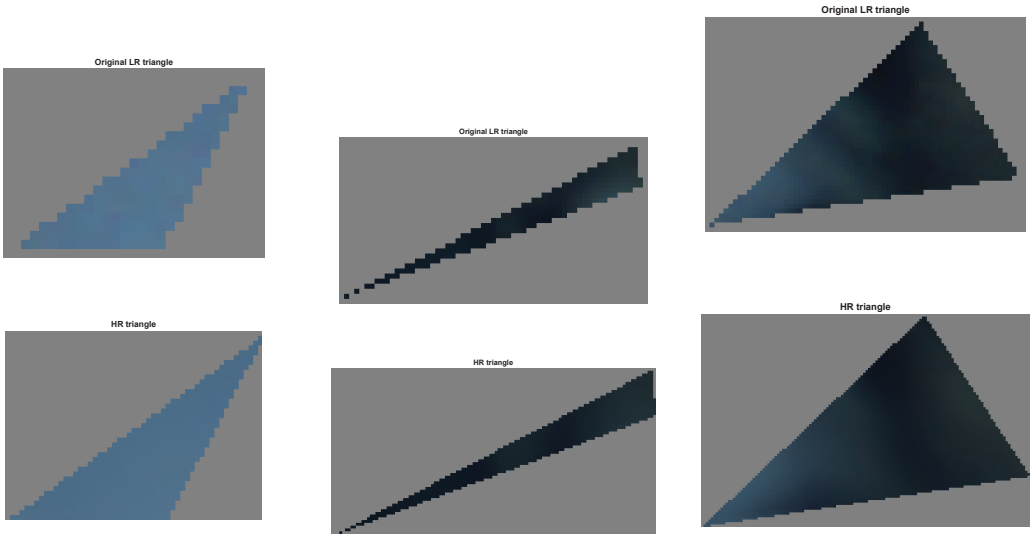
(a) image4 (b) image5 (c) image6

Fig. 5.19: Results 2



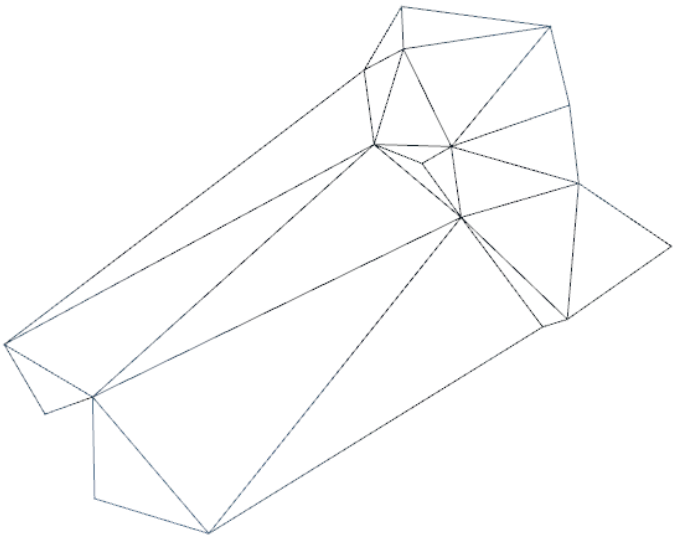
(a) image7 (b) image8 (c) image9

Fig. 5.20: Results 3

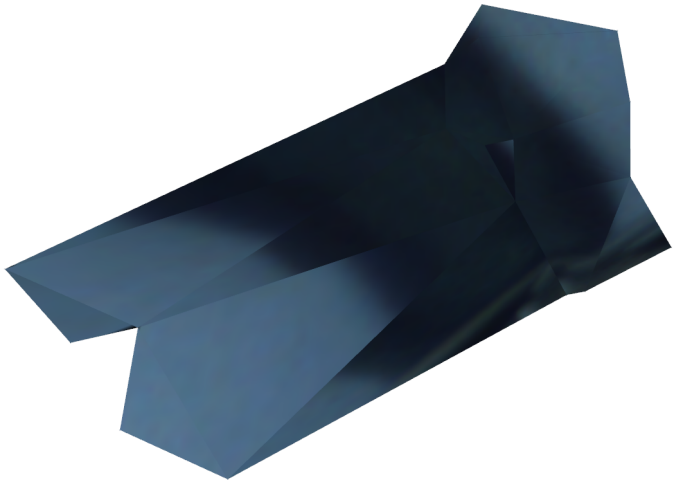


(a) image10 (b) image11 (c) image12

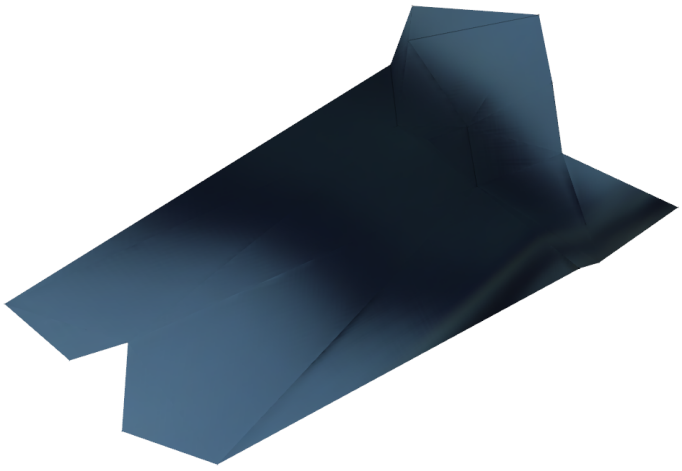
Fig. 5.21: Results 4



(a) Wireframe



(b) Textured with LR texture



(c) Textured with HR texture

Fig. 5.22: Final output TDSM for small portion

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

In this chapter, a summary of the main contributions of this dissertation is presented along with the direction for future research. The overall goal of this dissertation is to provide an algorithm to create scientifically accurate TDSM for a large scale with low-cost equipment. The introductory Chapters 1 and 2 provided the background information, the history of creating TDSM using texel images, and the necessary topics for readers who are not familiar with the topics that are discussed in the subsequent chapters. An overall algorithm was given in a flowchart to inform readers about the contribution provided by this dissertation and the improvement over the existing algorithm.

Chapter 3 presents the comprehensive background and theory of Bundle Adjustment and introduced the idea of Streaming Bundle Adjustment that is applied to create TDSM for large-scale applications. The advantage of a texel camera in creating scientifically accurate TDSM using Conventional Bundle Adjustment was evident from the past works. The registration result that was obtained using SBA was comparatively similar both visually and numerically to the CBA. The SBA was shown to be robust and the advantages of memory consumption and the time it takes to complete the overall algorithm make it very suitable to be applied to any large-scale application. The unique cost function comprising image projections and the range data makes optimization more robust. The course measurement from a low-cost inertial system is a perfect seed needed for optimization. The use of a bistatic texel camera described in Chapter 2 is easy to implement in the hardware compared to a coboresighted sensor used in the past. Even with the complication of incorporating a bistatic sensor in the cost function, a closed-form expression was found for the Jacobian calculation, which is more precise than numerical methods.

The concept of using optimal triangle to texture and a way of packing is introduced in chapter 4. When a TIN is created using all optimized 3D points, the next step is to texture such regions with the best texture available. Many criteria to find the best texture triangle was discussed and a way to calculate a metric by combining all criteria were presented. This chapter presents a promising result of using the best texture according to the position and orientation of the triangles in a TIN and the texture images. Since such texture triangle is not continuous, an efficient way to store such triangles was presented. An iterative algorithm was implemented to pack all triangles into an image canvas. A significant image size saving was accomplished by storing the best triangle compared to saving all candidate full-sized images. By splitting the image canvas into three regions, a further reduction in size was accomplished.

Finally, a concept of Superresolution was introduced in Chapter 5. Instead of using the best triangle like Chapter 4, an SR algorithm makes use of all the available texture triangles to generate a super-resolved texture triangle. A super-resolved texture triangle has the same shape as the best triangle but more resolution and the pixels were contributed by all available texture triangles. A novel concept of applying SR in a triangle-based fashion using a well-known Wiener filter was discussed. As this chapter is only a preliminary study of SR, a simple PSF was considered for the imaging system and a simple covariance model was assumed between two HR pixels. Superresolution was applied to a small part of the TDSM and showed improvement in triangle texture and overall TDSM.

6.2 Future Work

There are many improvements that can be applied in order to improve the quality of TDSMs. In this work, we compared the quality of TDSMs using SBA with the TDSMs computed using CBA. With the help of ground truth, it can be compared with the actual terrain. In the future, such data acquisition can be performed.

The current work assumes all the lidar points in a texel image are good data with small errors. If a bad point exists in the texel image, the optimization routine tries to optimize the bad point which affects the registration result. In the future, such outliers can be removed

using RANSAC which can greatly improve the registration results. Additionally, a Kalman filter can be applied if the pairwise registration failed because of a lack of features in the texel images.

Regarding improving the texture of the TDSM, a better algorithm that does the 3D surface reconstruction instead of 2D Delaunay triangulation can be used to create TIN. Furthermore, a complex imaging model can be used to estimate the HR triangle.

Future work on packing the optimal triangle includes rotating the triangle texture to further make the final image more compact. The algorithm in this work can easily be extended to multiple return lidar to create Digital Elevation Maps along with surface maps.

REFERENCES

[1] Gomez, C., Hayakawa, Y., and Obanawa, H., "A study of Japanese landscapes using structure from motion derived DSMs and DEMs based on historical aerial photographs: New opportunities for vegetation monitoring and diachronic geomorphology," *Geomorphology*, Vol. 242, 2015, pp. 11 – 20, Geomorphology in the Geocomputing Landscape: GIS, DEMs, Spatial Analysis and statistics.

[2] Anderson, K. and Gaston, K., "Lightweight unmanned aerial vehicles will revolutionize spatial ecology," Vol. 11, 03 2013, pp. 138–146.

[3] Remondino, F., "Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning," *Remote Sensing*, Vol. 3, No. 6, 2011, pp. 1104–1138.

[4] Lim, K., Treitz, P., Wulder, M., St-Onge, B., and Flood, M., "LiDAR remote sensing of forest structure," *Progress in Physical Geography*, Vol. 27, No. 1, 2003, pp. 88–106.

[5] Jain, M. K. and Singh, V. P., "DEM-based modelling of surface runoff using diffusion wave equation," *Journal of Hydrology*, Vol. 302, No. 1, 2005, pp. 107 – 126.

[6] Bisson, M., Behncke, B., Fornaciai, A., and Neri, M., "LiDAR-based digital terrain analysis of an area exposed to the risk of lava flow invasion: the Zafferana Etnea territory, Mt. Etna (Italy)," *Natural Hazards*, Vol. 50, No. 2, Aug 2009, pp. 321–334.

[7] Hartley, R. and Zisserman, A., *Multiple view geometry in computer vision*, Cambridge University Press, 2003.

[8] Elstrom, M. D. and Smith, P. W., "Stereo-Based Registration of Multi-Sensor Imagery for Enhanced Visualization of Remote Environments," Vol. 3, IEEE, Detroit, MI, May 1999, pp. 1948–1953.

[9] Skolnik, M. I., "Introduction to radar," *Radar Handbook*, 1962, pp. 2.

[10] Guenther, G. C., "Airborne Laser Hydrography: System Design and Performance Factors," 1985.

[11] Zhang, Q. and Pless, R., "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 3, Sept 2004, pp. 2301–2306 vol.3.

[12] Ding, M., Lyngbaek, K., and Zakhor, A., "Automatic registration of aerial imagery with untextured 3D LiDAR models," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[13] Hahne, U. and Alexa, M., "Combining Time-of-Flight Depth and Stereo Images Without Accurate Extrinsic Calibration," *Int. J. Intell. Syst. Technol. Appl.*, Vol. 5, No. 3/4, Nov. 2008, pp. 325–333.

[14] Zhang, J., Wang, L. H., Li, D. X., and Zhang, M., "High quality depth maps from stereo matching and ToF camera," *Soft Computing and Pattern Recognition (SoCPaR), 2011 International Conference of*, Oct 2011, pp. 68–72.

[15] Stamos, I. and Allen, P. K., "Integration of range and image sensing for photorealistic 3D modeling," *in Proc. IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 1435–1440.

[16] Ko, R. K., Nishino, K., Zhang, Z., and Ikeuchi, K., "Simultaneous 2D images and 3D geometric model registration for texture mapping utilizing reflectance attribute," *In Proceedings of Fifth Asian Conference on Computer Vision*, 2002, pp. 99–106.

[17] Umeda, K., Godin, G., and Rioux, M., "Registration of range and color images using gradient constraints and range intensity images," *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, Vol. 3, Aug 2004, pp. 12–15 Vol.3.

[18] Boldt, B. M., Budge, S. E., Pack, R. T., and Israelsen, P. D., "A Handheld Texel Camera for Acquiring Near-Instantaneous 3D Images," *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, 2007, pp. 953–957.

[19] Budge, S. E. and Badamikar, N. S., "Calibration Method for Texel Images Created from Fused Lidar and Digital Camera Images," Vol. 52, No. 10, Oct. 2013, pp. 103101–103101.

[20] Khatiwada, B. and Budge, S. E., "Three-dimensional image reconstruction using bundle adjustment applied to multiple texel images," *Laser Radar Technology and Applications XXI*, edited by M. D. Turner and G. W. Kamerman, Vol. 9832, Baltimore, Maryland, USA, May 2016, pp. 98320S–98320S–8.

[21] Bybee, T. C. and Budge, S. E., "Textured digital elevation model formation from low-cost UAV ladar/digital image data," *Laser Radar Technology and Applications XX*, edited by M. D. Turner and G. W. Kamerman, Vol. 9465, Baltimore, Maryland, USA, May 2015, pp. 94650H–94650H–12.

[22] Hamilton, W. R., *Elements of quaternions*, Longmans, Green, & Company, 1866.

[23] Shoemake, K., "Animating rotation with quaternion curves," *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 245–254.

[24] Henderson, D., "Shuttle Program. Euler angles, quaternions, and transformation matrices working relationships," Tech. rep., 1977.

[25] Heikkila, J. and Silvén, O., "Calibration procedure for short focal length off-the-shelf CCD cameras," *Proceedings - International Conference on Pattern Recognition*, Vol. 1, 1996, pp. 166–170.

[26] Http://www.vision.caltech.edu/bouguetj/calib_doc/, "Camera Calibration Toolbox for MATLAB," .

[27] Welch, T. C., *Algorithms for the Calibration and Correction of Texel Images Using Inertial Measurement Updates*, Master's thesis, Utah State University, 2021.

[28] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L., "Speeded-up robust features (SURF)," *Computer vision and image understanding*, Vol. 110, No. 3, 2008, pp. 346–359.

[29] He, F. and Habib, A., "Automated relative orientation of UAV-based imagery in the presence of prior information for the flight trajectory," *Photogrammetric Engineering & Remote Sensing*, Vol. 82, No. 11, 2016, pp. 879–891.

[30] Trevor Welch, Calvin Coopmans, S. E. B., "Development of a small unmanned aerial system-mounted texel camera," *Laser Radar Technology and Applications XXIV*, edited by M. D. Turner and G. W. Kamerman, Vol. 11005, Baltimore, Maryland, USA, May 2019, p. 110050F.

[31] Khatiwada, B. and Budge, S. E., "Improved TDEM formation using fused ladar/digital imagery from a low-cost small UAV," *Laser Radar Technology and Applications XXII*, edited by M. D. Turner and G. W. Kamerman, Vol. 10191, Anaheim, California, USA, May 2017, pp. 1019105–1019105–9.

[32] Bybee, T. C. and Budge, S. E., "Method for 3-D Scene Reconstruction Using Fused LiDAR and Imagery From a Texel Camera," *IEEE Trans. Geosci. Remote Sens.*, Vol. 57, No. 11, Nov. 2019, pp. 8879–8889, doi:10.1109/TGRS.2019.2923551.

[33] Agarwal, S., Snavely, N., Seitz, S. M., and Szeliski, R., "Bundle adjustment in the large," *European conference on computer vision*, Springer, 2010, pp. 29–42.

[34] Konolige, K. and Garage, W., "Sparse Sparse Bundle Adjustment." *BMVC*, Vol. 10, Citeseer, 2010, pp. 102–1.

[35] Byröd, M. and Åström, K., "Conjugate gradient bundle adjustment," *European Conference on Computer Vision*, Springer, 2010, pp. 114–127.

[36] Jeong, Y., Nister, D., Steedly, D., Szeliski, R., and Kweon, I.-S., "Pushing the Envelope of Modern Methods for Bundle Adjustment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 8, 2012, pp. 1605–1617.

[37] Ni, K., Steedly, D., and Dellaert, F., "Out-of-Core Bundle Adjustment for Large-Scale 3D Reconstruction," *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[38] Shum, H.-Y., Ke, Q., and Zhang, Z., "Efficient bundle adjustment with virtual key frames: a hierarchical approach to multi-frame structure from motion," *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, Vol. 2, 1999, pp. 538–543 Vol. 2.

[39] Steedly, D. and Essa, I., "Propagation of innovative information in non-linear least-squares structure from motion," *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Vol. 2, 2001, pp. 223–229 vol.2.

[40] Zhang, Z. and Shan, Y., "Incremental motion estimation through modified bundle adjustment," *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, Vol. 2, 2003, pp. II–343.

[41] Engels, C., Stewénius, H., and Nistér, D., "Bundle adjustment rules," *Photogrammetric computer vision*, Vol. 2, No. 32, 2006.

[42] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P., "Real time localization and 3d reconstruction," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 1, IEEE, 2006, pp. 363–370.

[43] Indelman, V., Roberts, R., and Dellaert, F., "Incremental light bundle adjustment for structure from motion and robotics," *Robotics and Autonomous Systems*, Vol. 70, 2015, pp. 63–82.

[44] Malik, P. D. T., Debevec, P. E., and Taylor, C. J., "Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach," *Proc. ACM SIGGraph*, Vol. 96, 1996, pp. 11–20.

[45] Alshawabkeh, Y., "Integration of laser scanning and photogrammetry for heritage documentation," 2006.

[46] Hanusch, T., "A new texture mapping algorithm for photorealistic reconstruction of 3D objects," *Proc XXI ISPRS Congress*, 2008, pp. 699–706.

[47] El-Hakim, S., Gonzo, L., Picard, M., Girardi, S., and Simoni, A., "Visualization of Frescoed surfaces: Buonconsiglio Castle–Aquila Tower,"cycle of the months"," *Proc. of Int. Workshop on Visualization and Animation of Realty-based 3D Models, Tarasp-Vulpera, Switzerland*, 2003.

[48] Frueh, C., Sammon, R., and Zakhor, A., "Automated texture mapping of 3D city models with oblique aerial imagery," *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, IEEE, 2004, pp. 396–403.

[49] Alj, Y., Boisson, G., Bordes, P., Pressigout, M., and Morin, L., "Space carving mvd sequences for modeling natural 3D scenes," *Three-Dimensional Image Processing (3DIP) and Applications II*, Vol. 8290, International Society for Optics and Photonics, 2012, p. 829005.

[50] Lempitsky, V. and Ivanov, D., "Seamless mosaicing of image-based texture maps," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–6.

[51] Alj, Y., Boisson, G., Bordes, P., Pressigout, M., and Morin, L., "Multi-texturing 3D models: how to choose the best texture?" *2012 International Conference on 3D Imaging (IC3D)*, IEEE, 2012, pp. 1–8.

[52] Lai, J.-Y., Wu, T.-C., Phothong, W., Wang, D. W., Liao, C.-Y., and Lee, J.-Y., "A high-resolution texture mapping technique for 3D textured model," *Applied Sciences*, Vol. 8, No. 11, 2018, pp. 2228.

[53] Bikalpa Khatiwada, S. E. B., "Texturing of digital surface maps (DSMs) by selecting the texture from multiple perspective texel swaths taken by a low-cost small unmanned aerial vehicle (UAV)," *Laser Radar Technology and Applications XXIV*, edited by M. D. Turner and G. W. Kamerman, Vol. 11005, Baltimore, Maryland, USA, May 2019, p. 110050G.

[54] Nasrollahi, K. and Moeslund, T. B., "Super-resolution: a comprehensive survey," *Machine vision and applications*, Vol. 25, No. 6, 2014, pp. 1423–1468.

[55] Akgun, T., Altunbasak, Y., and Mersereau, R., "Super-resolution reconstruction of hyperspectral images," *IEEE Transactions on Image Processing*, Vol. 14, No. 11, 2005, pp. 1860–1875.

[56] Gunturk, B. K., Altunbasak, Y., and Mersereau, R. M., "Multiframe resolution-enhancement methods for compressed video," *IEEE Signal Processing Letters*, Vol. 9, No. 6, 2002, pp. 170–174.

[57] Kennedy, J. A., Israel, O., Frenkel, A., Bar-Shalom, R., and Azhari, H., "Super-resolution in PET imaging," *IEEE transactions on medical imaging*, Vol. 25, No. 2, 2006, pp. 137–147.

[58] Li, F., Jia, X., and Fraser, D., "Universal HMT based super resolution for remote sensing images," *2008 15th IEEE International Conference on Image Processing*, IEEE, 2008, pp. 333–336.

[59] Huang, T. S. and Tsay, R. Y., "Multiple frame image restoration and registration," *Advances in Computer Vision and Image Processing*, Vol. 1, JAI, Greenwich, 1984, pp. 317–339.

[60] Irani, M. and Peleg, S., "Super resolution from image sequences," *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, Vol. 2, IEEE, 1990, pp. 115–120.

[61] Elad, M. and Hel-Or, Y., "A Fast Super-resolution Reconstruction Algorithm for Pure Translational Motion and Common Space-invariant Blur," *Trans. Img. Proc.*, Vol. 10, No. 8, Aug. 2001, pp. 1187–1193.

[62] Zhang, X., Lam, E. Y., Wu, E. X., and Wong, K. K., "Medical Imaging and Informatics," chap. Application of Tikhonov Regularization to Super-Resolution Reconstruction of Brain MRI Images, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 51–56.

[63] Freeman, W. T., Jones, T. R., and Pasztor, E. C., "Example-based super-resolution," *IEEE Computer graphics and Applications*, Vol. 22, No. 2, 2002, pp. 56–65.

[64] Freeman, W. T., Pasztor, E. C., and Carmichael, O. T., "Learning low-level vision," *International journal of computer vision*, Vol. 40, No. 1, 2000, pp. 25–47.

[65] Chang, H., Yeung, D.-Y., and Xiong, Y., "Super-resolution through neighbor embedding," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 1, IEEE, 2004, pp. I–I.

[66] Datsenko, D. and Elad, M., "Example-based single document image super-resolution: a global MAP approach with outlier rejection," *Multidimensional Systems and Signal Processing*, Vol. 18, No. 2, 2007, pp. 103–121.

[67] He, H. and Siu, W. C., "Single image super-resolution using Gaussian process regression," *CVPR 2011*, June 2011, pp. 449–456.

[68] Kim, K. I. and Kwon, Y., "Single-Image Super-Resolution Using Sparse Regression and Natural Image Prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 6, June 2010, pp. 1127–1133.

[69] Dong, C., Loy, C. C., He, K., and Tang, X., "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 38, No. 2, 2016, pp. 295–307.

[70] Kim, J., Lee, J. K., and Lee, K. M., "Accurate image super-resolution using very deep convolutional networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[71] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z., "Real-time single image and video super-resolution using an efficient subpixel convolutional neural network," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[72] Nguyen, N. and Milanfar, P., "A Wavelet-Based Interpolation-Restoration Method for Superresolution," Vol. 19, 07 2000, pp. 321–338.

[73] Seunghyeon Rhee, M. G. K., "Discrete cosine transform based regularized high-resolution image reconstruction algorithm," *Optical Engineering*, Vol. 38, 1999, pp. 38 – 38 – 9.

[74] Park, S. C., Park, M. K., and Kang, M. G., "Super-resolution image reconstruction: a technical overview," *IEEE Signal Processing Magazine*, Vol. 20, No. 3, 2003, pp. 21–36.

[75] Irani, M. and Peleg, S., "Improving Resolution by Image Registration," *CVGIP: Graph. Models Image Process.*, Vol. 53, No. 3, April 1991, pp. 231–239.

[76] Stark, H. and Oskoui, P., "High-resolution image recovery from image-plane arrays, using convex projections," *J. Opt. Soc. Am. A*, Vol. 6, No. 11, Nov 1989, pp. 1715–1726.

[77] Panda, S. S., Prasad, M. S. R. S., and Jena, G., "POCS Based Super-Resolution Image Reconstruction Using an Adaptive Regularization Parameter," *CoRR*, Vol. abs/1112.1484, 2011.

[78] Patti, A. J., Sezan, M. I., and Tekalp, A. M., "Superresolution video reconstruction with arbitrary sampling lattices and nonzero aperture time," *IEEE transactions on image processing*, Vol. 6, No. 8, 1997, pp. 1064–1076.

[79] Ng, M. K., Shen, H., Lam, E. Y., and Zhang, L., "A total variation regularization based super-resolution reconstruction algorithm for digital video," *EURASIP Journal on Advances in Signal Processing*, Vol. 2007, 2007, pp. 1–16.

[80] Farsiu, S., Robinson, M. D., Elad, M., and Milanfar, P., "Fast and robust multiframe super resolution," *IEEE Transactions on Image Processing*, Vol. 13, No. 10, Oct 2004, pp. 1327–1344.

[81] Shen, H., Zhang, L., Huang, B., and Li, P., "A MAP Approach for Joint Motion Estimation, Segmentation, and Super Resolution," *IEEE Transactions on Image Processing*, Vol. 16, No. 2, Feb 2007, pp. 479–490.

[82] Nguyen, N., Milanfar, P., and Golub, G., "A Computationally Efficient Superresolution Image Reconstruction Algorithm," *Trans. Img. Proc.*, Vol. 10, No. 4, April 2001, pp. 573–583.

[83] Schultz, R. and Stevenson, R., "Extraction of high-resolution frames from video sequences," *IEEE Transactions on Image Processing*, Vol. 5, No. 6, 1996, pp. 996–1011.

[84] Hardie, R., Barnard, K., and Armstrong, E., "Joint MAP registration and high-resolution image estimation using a sequence of undersampled images," *IEEE Transactions on Image Processing*, Vol. 6, No. 12, 1997, pp. 1621–1633.

[85] Pan, R. and Reeves, S. J., "Efficient Huber-Markov edge-preserving image restoration," *IEEE Transactions on Image Processing*, Vol. 15, No. 12, 2006, pp. 3728–3735.

[86] Clark, J., Palmer, M., and Lawrence, P., "A transformation method for the reconstruction of functions from nonuniformly spaced samples," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 33, No. 5, 1985, pp. 1151–1165.

[87] Hardie, R., "A Fast Image Super-Resolution Algorithm Using an Adaptive Wiener Filter," *IEEE Transactions on Image Processing*, Vol. 16, No. 12, Dec 2007, pp. 2953–2964.

[88] Narayanan, B., Hardie, R. C., Barner, K. E., and Shao, M., "A computationally efficient super-resolution algorithm for video processing using partition filters," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 17, No. 5, 2007, pp. 621–634.

CURRICULUM VITAE

Bikalpa Khatiwada

Education

- **PhD in Electrical Engineering (2014 - Present)**
  Utah State University, Logan, UT
- **Bachelor of Electronics and Communication Engineering (2008 - 2012)**
  Tribhuvan University, Institute of Engineering, Pulchowk, Nepal

Published Conference Papers

- Texturing of digital surface maps (DSMs) by selecting the texture from multiple perspective texel swaths taken by a low-cost small unmanned aerial vehicle (UAV), Bikalpa Khatiwada and Scott E. Budge, in *Laser Radar Technology and Applications XXIV*, 2019.
- "Super-resolution textured digital surface model formation using aerial texel images taken from a low-cost, small unmanned aerial system, Bikalpa Khatiwada and Scott E. Budge, in *Laser Radar Technology and Applications XXIII*, 2018.
- Improved TDEM formation using fused ladar/digital imagery from a low-cost small UAV, Bikalpa Khatiwada and Scott E. Budge, in *Laser Radar Technology and Applications XXII*, 2017.
- Three-dimensional image reconstruction using bundle adjustment applied to multiple texel images, Bikalpa Khatiwada and Scott E. Budge, in *Laser Radar Technology and Applications XXI*, 2016.