LIDAR SCENE RECONSTRUCTION USING IMAGERY AND FRAME OVERLAP FROM A TEXEL CAMERA

by

Blake Chamberlain

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

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Scott Budge, Ph.D. Major Professor	Todd Moon, Ph.D. Committee Member
Cal Coopmans, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2025

ABSTRACT

LiDAR Scene Reconstruction Using Imagery and Frame Overlap from a Texel Camera

by

Blake Chamberlain, Master of Science

Utah State University, 2025

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

A small unmanned aerial vehicle (sUAV) can be used to reconstruct a 3D scene by capturing frames consisting of LiDAR and aerial photography data, creating a textured digital surface model (TDSM) with the full LiDAR point cloud and an overlaid registered image. Forming a complete 3D scene using texel image data (fused LiDAR and digital image scans) from an entire flight can be computationally prohibitive on low-cost hardware, so a streaming bundle adjustment algorithm can be used to process the data using a sliding window. The streaming algorithm uses less memory and is faster than a full bundle adjustment. Depending on the flight pattern, matching points in the scene may be visible from frames which are not adjacent in time, so reconstructing a complete scene can take into account matching points from non-adjacent frames to better correct for error.

A modification to the existing streaming bundle adjustment algorithm is described which finds overlap in frames which are not adjacent in time to correct for error. The objective is to optimize the streaming bundle adjustment for data sets which have a large amount of Texel images which are not captured consecutively in time. This algorithm also addresses the "loop-closing" problem that occurs when the sensor returns to the starting point of a survey. Flight data from an sUAV using a sensor constructed with low cost, commercial off-the-shelf parts is used to demonstrate how 3D scene reconstruction using this algorithm can correct for errors compared with data gathered from a full-scale aircraft. Examples of the resulting TDSMs are presented. The presented research can be applied to a variety of applications utilizing Texel cameras for terrain reconstruction, such as using multiple sUAVs capturing the same area of terrain simultaneously, and allowing for accurate reconstruction without requiring a strictly defined flight path to capture Texel images consecutively.

(80 pages)

PUBLIC ABSTRACT

LiDAR Scene Reconstruction Using Imagery and Frame Overlap from a Texel Camera

Blake Chamberlain

A small unmanned aerial vehicle (sUAV) can be used to reconstruct a large area of terrain by capturing 3D scans, consisting of LiDAR point clouds and an overlaid image, and combining the scans together. Forming a complete 3D scene using texel image data (fused LiDAR and digital image scans) from an entire flight can be computationally prohibitive on low-cost hardware, so combining registering scans is done using a streaming method which processes the scans in consecutive chunks. Depending on the flight pattern, matching points in the scene may be visible from scans which were not captured around the same time, so to improve the accuracy of the final reconstruction, reconstructing a complete scene must take into overlapping scans from any time in the flight.

A modification is described which finds overlap in scans which are not adjacent in time to correct for error. This algorithm also addresses the "loop-closing" problem, which occurs when the sensor returns to the starting point of a survey. In this work, flight data gathered from an sUAV with low-cost, commercial off-the-shelf sensors is used to demonstrate how 3D scene reconstruction using this algorithm can correct for errors. Examples of the resulting TDSMs are presented. The presented research can be applied to a variety of applications utilizing Texel cameras for terrain reconstruction, such as using multiple sUAVs capturing the same area of terrain simultaneously, and allowing for accurate reconstruction without requiring a strictly defined flight path to capture Texel images consecutively.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

3D	Three-dimensional
BA	Bundle Adjustment
CBA	Conventional Bundle Adjustment
DSM	Digital Surface Model
ENU	East North Up
EO	Electro-Optical
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LM	Levenberg-Marquardt
NCC	Normalized Cross Correlation
ORB	Oriented FAST and Rotated BRIEF
RANSAC	Random Sample Consensus
SBA	Streaming Bundle Adjustment
sUAV	Small Unmanned Aerial Vehicle
SURF	Speeded Up Robust Features
TDSM	Textured Digital Surface Model
TIN	Triangulated Irregular Network
UAV	Unmanned Aerial Vehicle

CHAPTER 1

Introduction

Textured Digital Surface Models (TDSMs) are used in a wide variety of applications when large 3D scans of terrain are required. One method to reconstruct a 3D scene is to use a small unmanned aerial vehicle (sUAV) to capture texel image data, which consists of fused LiDAR and digital image scans representing an area of terrain. The texel image data is then processed and registered after the flight, resulting in a single TDSM of the entire terrain. Ensuring the TDSM is highly accurate to the true terrain is an important problem.

To obtain an accurate TDSM, full-scale aircraft can be used along with high precision hardware and navigation equipment to properly reference LiDAR points in the world space, but the hardware and capture costs associated with this method is prohibitive for many applications. Less accurate hardware and navigation systems available at a lower cost can be used on smaller aircraft to obtain TDSMs, and the acquired data can be processed and registered after the flight to correct for errors which arise from using low precision hardware.

Correcting for measurement error in post-processing has the trade-off of a higher computational cost after the data is acquired, compared to using more accurate and more expensive measurement systems. Using a large amount of acquired texel images for a single terrain scan, and correcting for all the measurement errors in this data to obtain a single accurate TDSM representing a large area of terrain, can be computationally prohibitive on low-cost hardware. The computation and memory requirements for the post-processing error correction can be reduced by using a streaming bundle adjustment algorithm, which processes the data using a sliding window of texel images. “Bundle adjustment” refers to jointly optimizing the data. A bundle consists of a number of texel images grouped together, and a nonlinear optimization is performed to reduce the error present in this set of texel images. The streaming method for bundle adjustment, described later, groups texel images for error correction based on adjacency in time.

One weakness with the streaming algorithm can arise from the flight pattern when acquiring texel images. Depending on the flight pattern of the sUAV, there may be significant overlap present from texel images which are not adjacent in time, such as when the flight pattern returns to the starting point of the survey or loops back over a previously scanned area of terrain. The streaming algorithm as described [1] does not account for such overlap when it occurs. From this limitation, the algorithm is not using data which could help to improve the accuracy of the registration, and can result in uncorrected error from the loop-closing problem that occurs when the sensor returns to the starting point of a survey.

A modification to TDSM reconstruction using the streaming bundle adjustment algorithm was developed in this work. First, the post-processing step of TDSM reconstruction is modified to detect when overlap occurs between scans outside of the sliding window of neighboring texel images in the streaming algorithm. Next, the streaming bundle adjustment is augmented with the overlapping scans so that measurement errors can be corrected for, and the resulting TDSM is more accurate to the true terrain.

A method for efficient overlap detection in a large number of texel images was developed. This uses a broad search over every texel image in the flight to determine at which times in the flight pattern the sUAV acquired data from the same area of terrain. The streaming bundle adjustment algorithm was modified to use the detected overlap.

The method for augmenting the streaming bundle adjustment was compared to the streaming bundle adjustment, to determine the improvement in the resulting TDSM. Real-world flight data collected from a sUAV was used to measure and compare the effectiveness of each method. Improvements in the resulting TDSMs were found when compared with the streaming algorithm. The parameters for using the detected overlap can be adjusted and optimized for the flight pattern to create an accurate and efficient TDSM registration.

1.1 Point Set Registration

Registering multiple sets of 3D lidar points into a single scene is a widely researched problem. Some common algorithms to register 3D lidar points are Iterative Closest Point (ICP) and Coherent Point Drift (CPD) [2–4]. Using these algorithms, the poses for each lidar point set are adjusted to reduce errors and result in a more accurate final representation. When using ICP and CPD, the algorithms as described use only the 3D point clouds as input to the registration algorithms.

When registering 3D point sets gathered from an unmanned aerial or ground vehicle, positioning errors in the acquired data can greatly affect the accuracy of the registration and mapping. With sensors that determine the position of an unmanned vehicle during data acquisition, such as an inertial measurement unit (IMU) or a global positioning system (GPS) sensor, errors can accumulate over time and greatly affect the results of the final registration. Returning to the starting point of a survey requires the accumulation of these errors to be corrected for, so that the different areas of the scan can be accurately registered together and thus create a TDSM which reflects the true terrain. Correcting for the accumulation of these errors is a previously researched topic [5–7].

For point set registration in specialized applications, using techniques which utilize additional data available, such as image data associated with the point clouds, can help to improve the accuracy of the resulting point cloud [8–10]. This concept can be applied to terrain reconstruction using texel images. A single texel image contains an electro-optical (EO) image, taken from an optical camera, as well as a 3D point cloud, constructed from range data acquired from a lidar sensor. The 3D point cloud and 2D image can be fused to reference each 3D point in the point cloud to the corresponding (U, V) coordinate in the normalized image plane on the EO image. The 3D point cloud and the 2D EO image are fused into a single texel image. This is sometimes referred to as a 2.5D image, as it is a 3D image from a single point of view.

With texel images, both the 3D point cloud and 2D EO image data can be used to increase the accuracy of the final registration of multiple sets of 3D points. Key features present in the images can be detected and extracted using Scale-Invariant Feature Transform (SIFT) [11] and matched to corresponding key features in neighboring images to improve point cloud registration [12, 13].

1.2 Bundle Adjustment

A method for using lidar data and imagery in texel images to correct for measurement error and form a single TDSM was implemented and analyzed by Bybee et al [14, 15]. An iterative objective function for texel frame triangulation is used in post-processing after the flight data is captured. During data acquisition for a texel image, the camera pose is recorded from an inertial navigation system (INS), which allows the 3D points to be placed in the world frame and referenced to points acquired from other texel images. The objective function combines the measured range error and point projection error between captured images, along with the point projections into neighboring images found using image correlation, and uses bundle adjustment to correct the poses.

Bundle adjustment is a non-linear least-squares optimization problem which, in Bybee's method, applies the Levenberg-Marquardt algorithm to efficiently optimize the camera poses and find the maximum likelihood estimate. The corrected poses resulting from the Levenberg-Marquardt algorithm are then used to register the lidar points from multiple texel images into a single set of points, and the EO image data from is then used to form a single orthorectified image. This image is overlaid on a triangulated network of the corrected lidar points to form a single TDSM.

The four basic steps for bundle adjustment and texel image registration are outlined by Bybee:

1. Estimate pairwise homographies between neighboring texel images.

2. Create the digital surface model (DSM) texture. The pairwise homographies are used to construct a registered image (texture) and the LiDAR-to-texture mapping.

3. Select common LiDAR projection points. Pairwise homographies are used to estimate the projection of each LiDAR measurement into the neighboring texel images, and image correlation is used to refine this estimate.

4. Bundle adjustment of camera poses and 3D information. The projections and LiDAR measurements are used to triangulate the point cloud. Triangulation is done by forming an objective function involving the LiDAR range measurements, their calibrated projections into the associated texel imagery, and their projection into neighboring texel imagery.

During bundle adjustment, the poses and points are corrected based on the variance parameters of the measurement equipment, resulting in a TDSM which more accurately fits the true terrain. This method of bundle adjustment was found to be efficient and capable of correcting for errors present in texel images.

Processing very large sets of data in a single bundle adjustment has a large time and storage complexity which makes it infeasible [16]. For reconstructing a TDSM over a wide area, using data from a long flight with a large number of captured texel images, the processing limitations for single bundle adjustment make TDSM reconstruction unreasonable on most hardware.

1.3 Streaming Bundle Adjustment

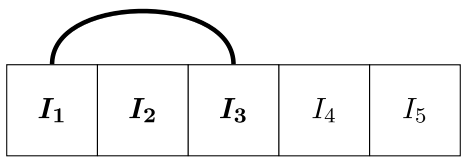
To lower the computation and memory requirements for bundle adjustment, a streaming bundle adjustment algorithm was developed by Khatiwada [1]. Instead of using all texel images present in the entire data set, it modifies the bundle adjustment algorithm to use a sliding window of texel images and corrects pose and point error over multiple iterations, eventually covering the entire data set. This limits the amount of data in a single bundle adjustment, using less memory and lowering runtime for TDSM reconstruction on large sets of data. The modification to use a sliding window can be made because the bundle adjustment for a single texel image's pose and 3D points is influenced by the point projections in the objective function. For a set of texel images which were captured consecutively, the bundle adjustment can be limited to only the adjacent texel images. Other non-adjacent texel images do not cover the same area of terrain, and can be left out from the bundle adjustment iteration.

An example of the sliding window is illustrated in Figure 1.1, where I_j denotes the j^{th} texel image in a data set. As Figure 1.1a shows, the sliding window performs bundle adjustment on the first three texel images in the data set, while not using I_4 and I_5 in the bundle adjustment. After this bundle adjustment is complete, the sliding window moves to remove I_1 and add I_4, as seen in Figure 1.1b, and another bundle adjustment is performed. This continues until the entire data set is covered.

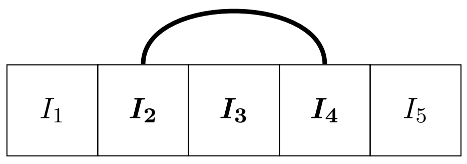
The streaming algorithm, as defined by Khatiwada, uses a sliding window which is only based on texel images which are time-adjacent, i.e. captured consecutively in time. This assumes that the dataset is arranged such that time-adjacent texel images share a number of visible points. For a typical flight from a sUAV, the flight velocity and capture height are set so that this assumption holds.

No consideration is taken to account for texel images which are not adjacent in time but capture the same area of terrain, such as when the sUAV returns to the starting point after capturing the flight, or flies over a previously captured area of terrain. Depending on the flight pattern, there may be significant overlap in a large number of texel images which are not time-adjacent. If this is not taken into account, and a single TDSM is created from a large number of frames using this streaming algorithm, the resulting TDSM can contain significant errors from GPS drift over time or other errors that accumulate in the intermediate texel images.

Adjusting the streaming bundle adjustment algorithm to account for such overlap is the focus of this research. The contribution of this research is to build on prior research which uses the streaming bundle algorithm, by optimizing the terrain reconstruction when using



(a) Sliding window for the first bundle adjustment.



(b) Sliding window for the second bundle adjustment.

Fig. 1.1: A demonstration of bundle adjustment using a sliding window.

data sets which include a large amount of non-adjacent texel images that cover the same area of terrain. A method of detecting overlap between texel images in large sets of data was developed, and this data was used to augment the streaming bundle adjustment. This keeps the advantages of the streaming bundle adjustment, in limiting the size of a single bundle adjustment to make processing feasible on low-cost hardware, while improving upon it to use additional data present in the flight during each bundle adjustment iteration to create a more accurate TDSM. The augmented streaming bundle adjustment algorithm developed in this research can be used to improve the accuracy of TDSM reconstruction from texel image data sets, such as data acquired from multiple sUAVs capturing the same area of terrain simultaneously, or data from any flight pattern with a single sUAV which captures texel images of the same area of terrain multiple times.

Chapter 2 describes the background necessary for this research. The methods developed are presented in Chapter 3. The results from testing this research are described in Chapter 4. Finally, Chapter 5 presents a conclusion and possible directions for future research building upon this work. The research in this thesis was presented in part at the 2024 SPIE Defense and Commercial Sensing conference [17].

CHAPTER 2

Background

2.1 Texel Camera and Texel Images

This research includes significant work developed prior by the Center for Advanced Imaging LiDAR CAIL at Utah State University. This chapter provides necessary background for developed research, and sections 2.1, 2.2, 2.3, 2.4, and 2.5 describe research developed prior, at the Center for Advanced Imaging LiDAR (CAIL) at Utah State University.

A texel camera is an instrument developed at CAIL, which consists of a lidar range sensor and an optical camera [18]. An example of a texel camera is shown in Figure 2.1. In addition to the optical camera and lidar sensor, it has an INS to capture the pose, and a TX2 processor to run the data acquisition software, which are visible in Figure 2.1.

The texel camera constructs a 3D point cloud from range data acquired from a lidar sensor, and acquires an electro-optical (EO) image. The 3D point cloud, consisting of a set of x, y, and z data points, and EO image are then fused into a single texel image by assigning (U, V) coordinate values to the lidar points. The (U, V) coordinates map the lidar points to locations in the EO image. The set of this data can then be used to generate a triangulated image network (TIN) and overlay the image, to create a single 2.5D textured image.

For this research, the texel camera was mounted to a small unmanned aerial vehicle (sUAV) to test on real-world data. To acquire data, the sUAV is flown over terrain while the texel camera captures texel images at a defined interval. Figure 2.2(a) shows the drone with texel camera payload, and Figure 2.2(b) shows the drone airborne, while capturing data in flight. The raw data acquired in flight from the texel camera sensors is saved to onboard memory. The raw data acquired is processed and fused into texel images after the

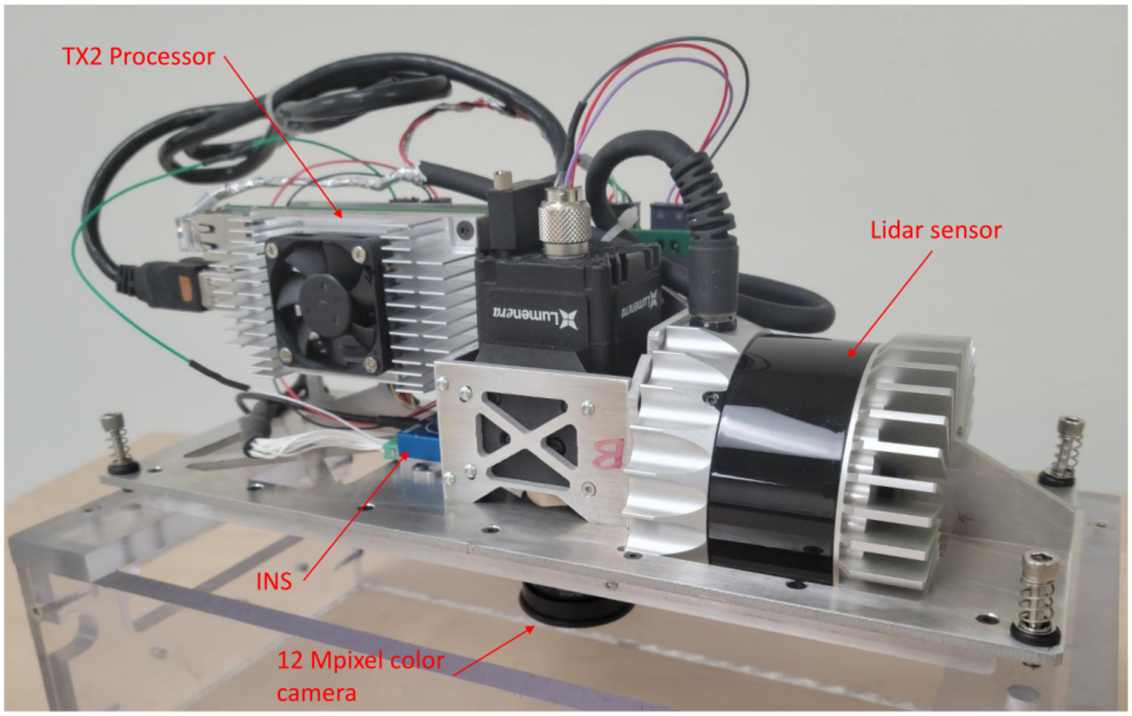


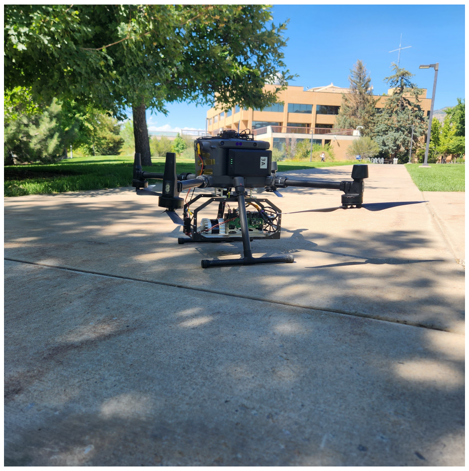
Fig. 2.1: The texel camera used in this research.

flight, using an image acquisition software developed by CAIL. When capturing real-world data, the texel camera is mounted to an sUAV developed and flown in collaboration with AggieAir [19].

An example of a texel image is shown in Figure 2.3. The corresponding point cloud is shown in Figure 2.4, illustrating how the 3D points are used in the texel image.

An example of a TDSM from multiple texel images registered together is shown in Figure 2.5. It can be seen that it covers a larger area than the example texel image from Figure 2.3. Figure 2.6 shows the triangulated wireframe for the TDSM, generated from the 3D points.

A bistatic texel camera was used. The lidar and optical camera are aligned side by side, so the center of projection for the lidar sensor and texel camera do not correspond to the same physical location. This results in different normalized planes for the lidar sensor and the optical camera. In this work, the normalized plane for the lidar sensor is referred to as the normalized lidar plane, and the normalized plane for the optical camera is referred


(a) sUAV on ground.

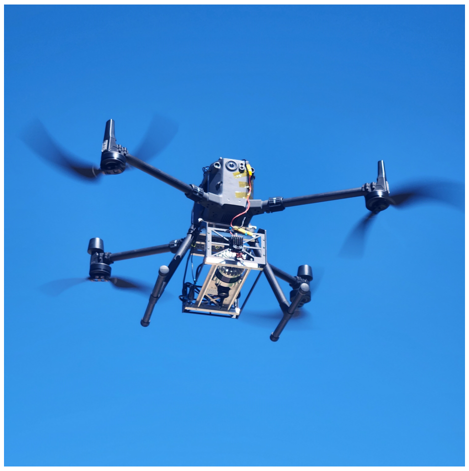

(b) sUAV in flight.

Fig. 2.2: The sUAV with texel camera payload.

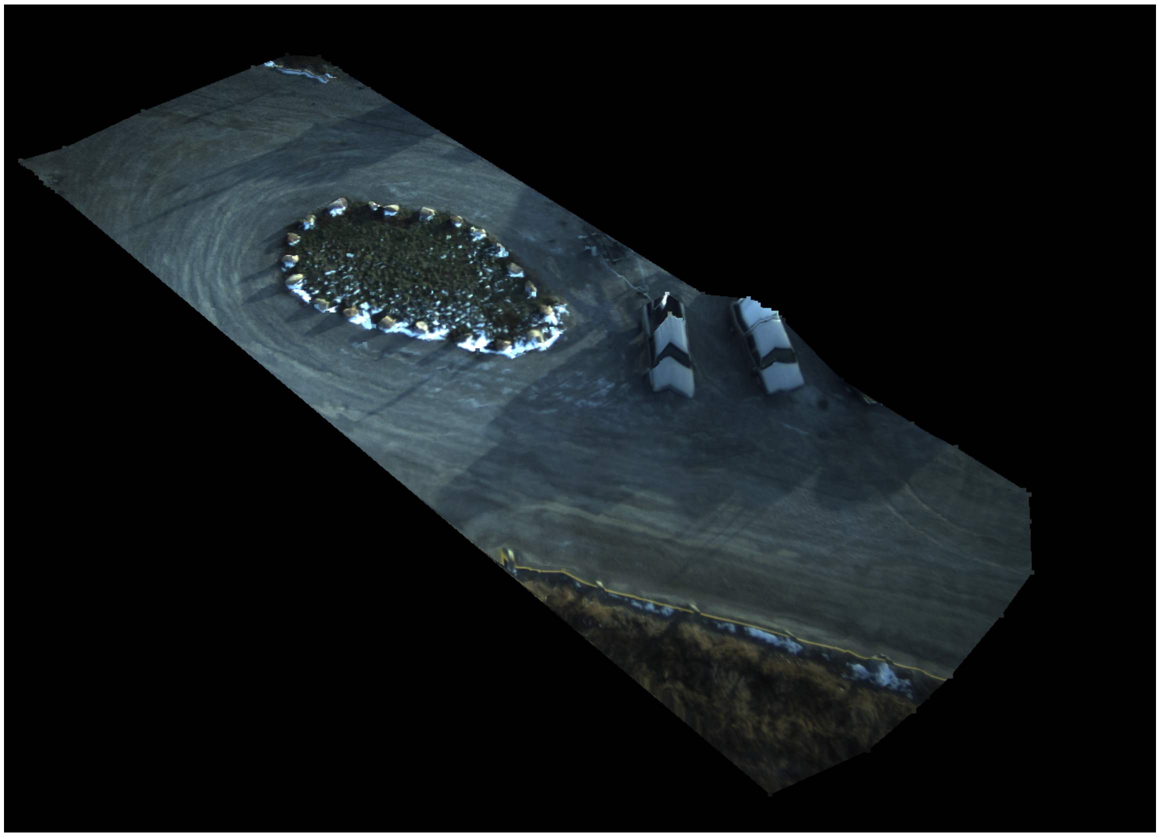


Fig. 2.3: Example of a texel image, with the optical image overlaid.

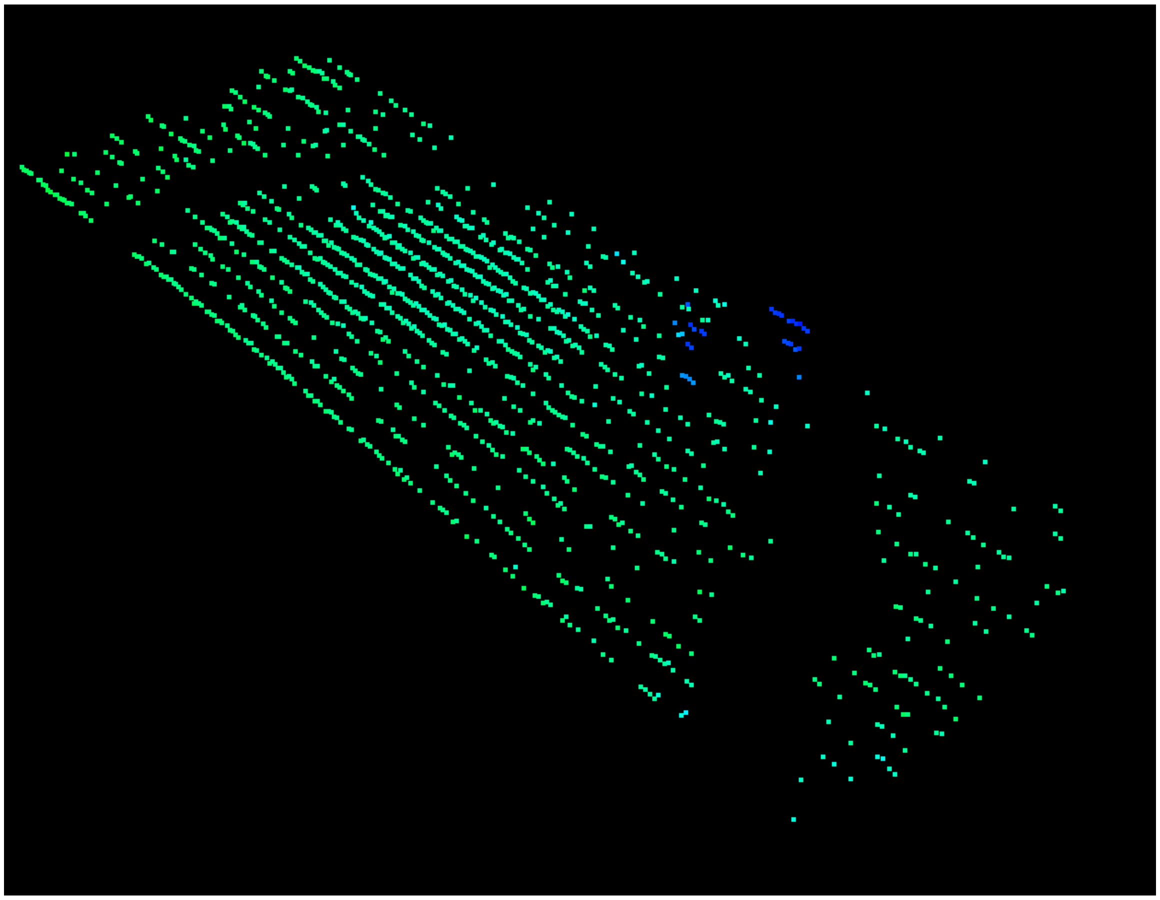


Fig. 2.4: 3D point cloud associated with texel image, generated from acquired lidar data. Each point is captured with a pulse from the lidar sensor. The colors of the points represent the range from the lidar sensor when the data was acquired. The blue points seen in the center of the image are at a higher elevation, and thus were closer to the position of the lidar sensor when the point cloud was captured.

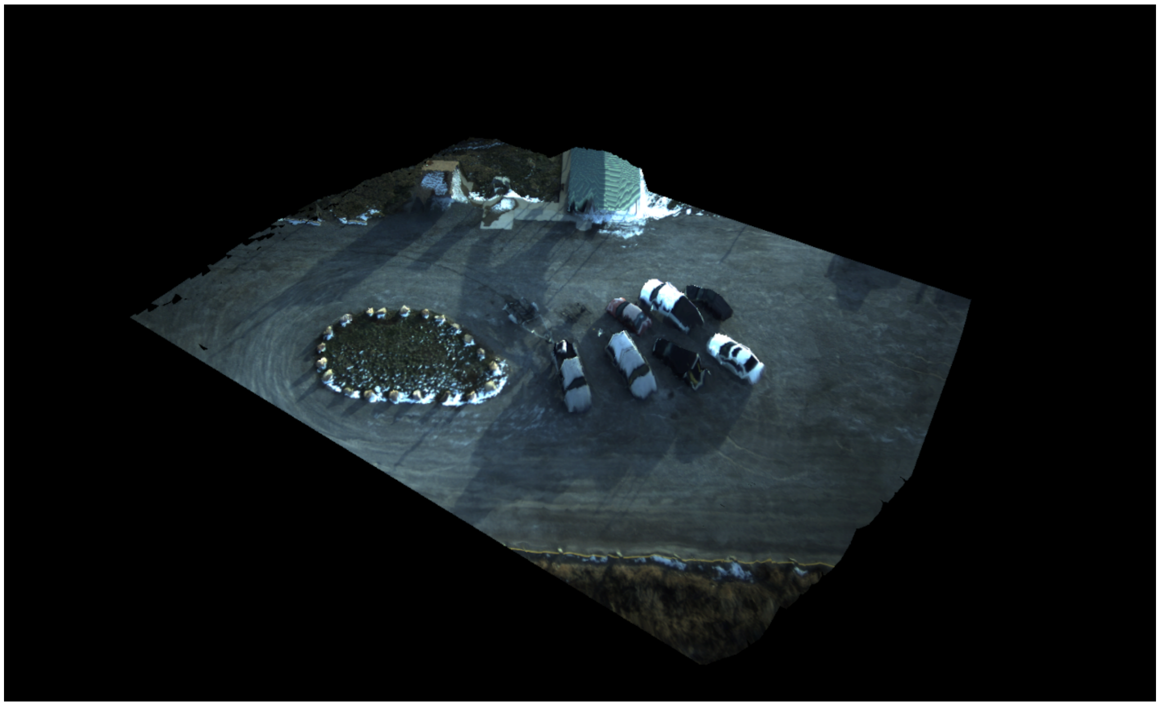


Fig. 2.5: Example of a TDSM, with the optical image overlaid.

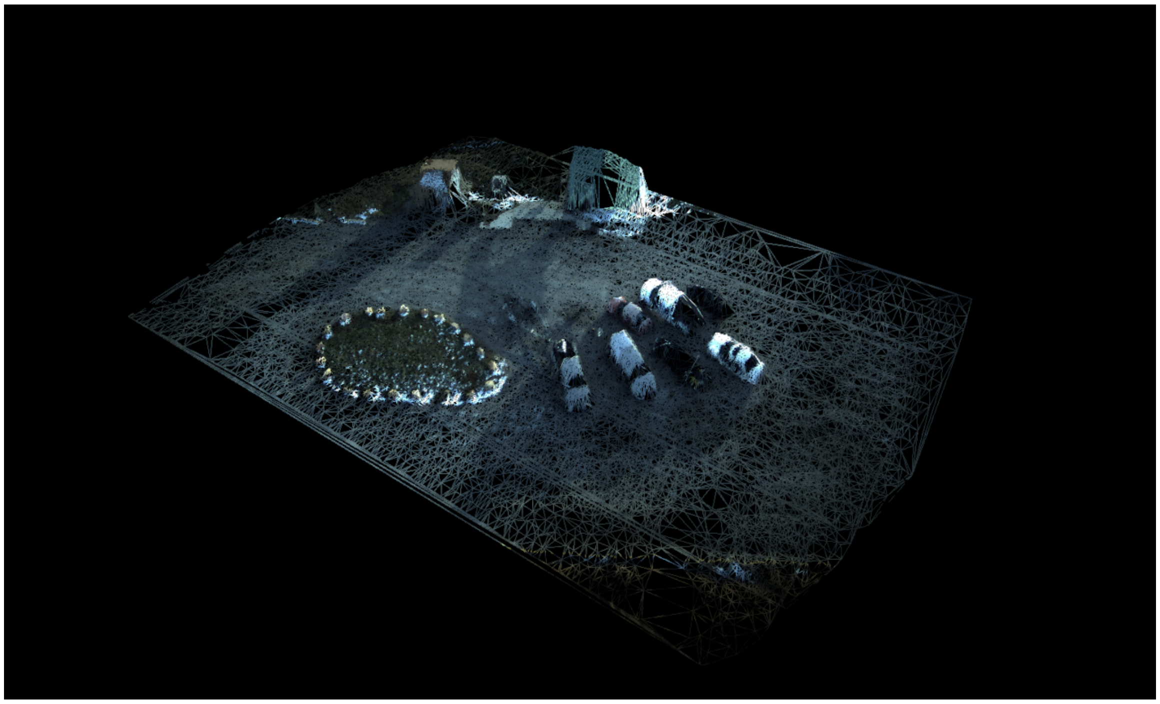


Fig. 2.6: 3D wireframe of TDSM, generated from acquired lidar data.

to as the normalized image plane.

Aligning the normalized lidar and image planes is required for the 3D points in a texel image to be projected onto the normalized image plane. This requires the lidar sensor and optical camera to be calibrated, which consists of three steps: calibrating the digital camera, aligning the lidar and digital camera, and estimating the mapping from the lidar to the camera.

First, the intrinsic calibration matrix, $\mathcal{K}$, is found for the camera. With this matrix, pixel values in the EO image can be converted to points in the normalized image plane, and vice versa. The optical camera's field of view is wider than the lidar sensor's field of view, so another necessary calibration is aligning the centers of projection in the lidar sensor and the digital camera. This calibration is completed during construction of the texel camera. The mapping for where the lidar points lie on the normalized image plane can then be calibrated and used.

2.2 Camera Pose and Point Projections

Along with the lidar sensor and optical camera, the texel camera used in this research contains an IMU to record the texel camera's pose. When a texel image is taken, the INS onboard provides an estimate of the texel camera's pose at that point in time. This measurement is recorded as a 3x3 rotation matrix R and a 3x1 translation matrix t, combined into a partitioned matrix $[R|t]$. Each camera pose is referenced to a common frame in the world coordinate system. The pose of the first texel image taken from the INS during data acquisition is set as the origin for the world coordinate system, given in East, North, Up (ENU) coordinates, with rotation and translation matrix set for this point as:

$$[R|t] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \tag{2.1}$$

The 3D points captured with the texel camera's lidar sensor are defined in coordinates in the camera frame, using the 3D data captured by the lidar sensor and the calibration of the lidar sensor to the camera. By referencing each camera pose to this origin in the world coordinate system, the 3D points from different texel images can be transformed into a common coordinate system, referred to as world coordinates. This transformation is done using the camera pose and INS.

Quaternions are used instead of the rotation and translation matrix because they are more computationally efficient and numerically stable. Quaternions also avoid gimbal lock that can occur when using Euler angles, when a degree of freedom is lost. Bybee et al [14,15] give the equations for finding quaternions from a rotation and translation matrix. Given a unit quaternion **q**, so that $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$, the rotation matrix R can be found:

$$[R] = \begin{bmatrix} 1 - \frac{2(q_2^2+q_3^2)}{q_0^2+q_1^2+q_2^2+q_3^2} & \frac{2(q_1q_2-q_0q_3)}{q_0^2+q_1^2+q_2^2+q_3^2} & \frac{2(q_1q_3+q_0q_2)}{q_0^2+q_1^2+q_2^2+q_3^2} \\ \frac{2(q_1q_2+q_0q_3)}{q_0^2+q_1^2+q_2^2+q_3^2} & 1 - \frac{2(q_1^2+q_3^2)}{q_0^2+q_1^2+q_2^2+q_3^2} & \frac{2(q_2q_3-q_0q_1)}{q_0^2+q_1^2+q_2^2+q_3^2} \\ \frac{2(q_1q_3-q_0q_2)}{q_0^2+q_1^2+q_2^2+q_3^2} & \frac{2(q_2q_3+q_0q_1)}{q_0^2+q_1^2+q_2^2+q_3^2} & 1 - \frac{2(q_1^2+q_2^2)}{q_0^2+q_1^2+q_2^2+q_3^2} \end{bmatrix}. \tag{2.2}$$

Using the rotation matrix R, with elements R_{ij} where i and j are the row and column values, the conversion from a rotation matrix to a quaternion can also be found:

$$\begin{aligned} q_0^2 &= \frac{1}{4}(1 + R_{11} + R_{22} + R_{33}), \\ q_1^2 &= \frac{1}{4}(1 + R_{11} - R_{22} - R_{33}), \\ q_2^2 &= \frac{1}{4}(1 - R_{11} + R_{22} - R_{33}), \\ q_3^2 &= \frac{1}{4}(1 - R_{11} - R_{22} + R_{33}). \end{aligned} \tag{2.3}$$

The quaternion can then be computed using:

$$\mathbf{q} = \frac{1}{4q_0}\begin{bmatrix} 4q_0^2 \\ R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} = \frac{1}{4q_1}\begin{bmatrix} R_{32} - R_{23} \\ 4q_1^2 \\ R_{12} + R_{21} \\ R_{13} + R_{31} \end{bmatrix} = \frac{1}{4q_2}\begin{bmatrix} R_{13} - R_{31} \\ R_{12} + R_{21} \\ 4q_2^2 \\ R_{23} - R32 \end{bmatrix} = \frac{1}{4q_3}\begin{bmatrix} R_{21} - R_{12} \\ R_{13} + R_{31} \\ R_{23} - R_{32} \\ 4q_3^2 \end{bmatrix}. \tag{2.4}$$

Each texel image j has its own set of 3D lidar points, denoted as I_j. Let $\mathbf{b}_i = [b_{ix}, b_{iy}, b_{iz}]^T$ be the i^{th} 3D point, inside 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 quaternion and translation for the camera pose for texel image j, where t_{jx}, t_{jy}, and t_{jz} are the translation. From this, we can define the following:

(x_{ij}, y_{ij}) as the true normalized image projection of the 3D point $\mathbf{b}_i$ in the j^{th} texel image, using the image correlation.

$(\hat{x}_{ij}, \hat{y}_{ij})$ as the estimated normalized image projection of $\mathbf{b}_i$ in the j^{th} texel image from the 3D points and pose.

λ_{ij} as the true range of $\mathbf{b}_i$ acquired from lidar data in the j^{th} texel image.

$\hat{\lambda}_i j$ as the estimated range of $\mathbf{b}_i$ using 3D points and pose in the j^{th} texel image.

The transformation of a point $\mathbf{b}_{ij}$ from the world frame to the j_{th} camera frame uses the quaternion poses, the camera-to-INS offset, and the camera-to-GPS antenna offset. When constructing the texel camera, the distance between the lidar sensor and optical camera, and the distance between the GPS antenna and camera are both measured. These can be considered part of the calibration parameters for the texel camera. Let d_{ant_x}, d_{ant_y}, d_{ant_z} be the measured distance between the optical camera and antenna in x, y, and z, and d_{cam_x}, d_{cam_y}, d_{cam_z} be the distance between the lidar sensor and camera in x, y, and z. Then, as Khatiwada shows [1], using the quaternion representation of a rotation matrix from equation 2.2, the transformation for a point from the world frame into the camera frame is given as:

$$[R_j] = \begin{bmatrix} 1 - \frac{2(q_{j2}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j2}-q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j1}q_{j3}+q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \\ \frac{2(q_{j1}q_{j2}+q_{j0}q_{j3})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1 - \frac{2(q_{j1}^2+q_{j3}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}-q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \\ \frac{2(q_{j1}q_{j3}-q_{j0}q_{j2})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & \frac{2(q_{j2}q_{j3}+q_{j0}q_{j1})}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} & 1 - \frac{2(q_{j1}^2+q_{j2}^2)}{q_{j0}^2+q_{j1}^2+q_{j2}^2+q_{j3}^2} \end{bmatrix}, \tag{2.5}$$

and

$$\hat{\mathcal{X}}_{ij} = \begin{bmatrix} \hat{\mathcal{X}}_{ij_x} \\ \hat{\mathcal{X}}_{ij_y} \\ \hat{\mathcal{X}}_{ij_z} \end{bmatrix} = [R_j]^T \begin{bmatrix} l_{i_x} - d_{ant_x} + d_{cam_x} - t_{j_x} \\ l_{i_y} - d_{ant_y} + d_{cam_y} - t_{j_y} \\ l_{i_z} - d_{ant_z} + d_{cam_z} - t_{j_z} \end{bmatrix}. \tag{2.6}$$

Let the point $\hat{\mathcal{X}}_{ij} = [\hat{\chi}_{ij_x}, \hat{\chi}_{ij_y}, \hat{\chi}_{ij_z}]^T$ be the transformation of $\mathbf{b}_i$ into the j^{th} camera frame. The point can be projected onto the j^{th} normalized camera plane, and the estimated range $\hat{\lambda}_{ij}$ can be obtained, using the equation:

$$\hat{\boldsymbol{X}}_{ij} = \begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \\ \hat{\lambda}_{ij} \end{bmatrix} = \begin{bmatrix} \frac{\hat{\chi}_{ij_x}}{\hat{\chi}_{ij_z}} \\ \frac{\hat{\chi}_{ij_y}}{\hat{\chi}_{ij_z}} \\ \sqrt{\hat{\chi}_{ij_x}^2 + \hat{\chi}_{ij_y}^2 + \hat{\chi}_{ij_z}^2} \end{bmatrix}. \tag{2.7}$$

The estimated point on the normalized image projection plane is $[\hat{x}_{ij}, \hat{y}_{ij}]^T$. When the estimated range, $\hat{\lambda}_{ij}$, is available, the projection-range representation can be written as:

$$\hat{\boldsymbol{X}}_{ij} = \begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \\ \hat{\lambda}_{ij} \end{bmatrix}. \tag{2.8}$$

The point $\hat{\boldsymbol{X}}_{ij}$ is used in finding the projection error and range error, which is utilized in bundle adjustment to optimize the poses.

For the bistatic texel camera used in this research, to create accurate point projections the physical separation between the lidar sensor, camera, and GPS antenna must be accounted for.

The camera pose quaternion $\mathbf{a}_j$ is used to transform a lidar point into the normalized image plane for the texel image j which captured the point. In order to find the necessary transformation to project a point $\mathbf{b}_i$ into a different texel image k, the pose and translation $\mathbf{a}_k$ is used. Since we are trying to find $\mathbf{a}_j$, we can't use it for finding the true point projections. Therefore, we use image processing methods to first estimate a homography between two images, and then the true point projections are found using Normalized Cross

Correlation (NCC).

2.3 Homography Estimation and Normalized Cross Correlation

Speeded Up Robust Features (SURF) is used to find the homography estimate [20]. A homography is a transformation matrix that describes the mapping between two images. SURF is a feature detector and descriptor commonly used in image processing applications. It is efficient, scale and rotation-invariant, robust, and is accurate when compared with other feature extraction algorithms. Another algorithm to find homographies, ORB (Oriented FAST and Rotated BRIEF) [21], was tested and resulted in similar homographies as SURF.

To find the homography between two texel images, the features in two neighboring texel images are extracted using SURF, and matching features between the images are found. The homography is computed from these matched features using Random Sample Consensus (RANSAC) [22], and a least-squares method to match the points and find the best estimate of the homography. If enough matching points between the two images were found, and the RANSAC confidence parameter meets a defined threshold, the homography is saved as an estimate which is assumed to be accurate.

Figure 2.7 illustrates the process for finding a homography estimate. This is a standard algorithm for point matching, commonly used to estimate homographies between two images.

Figure 2.8 shows the correspondences matched between two optical images, taken from two neighboring texel images. A single correspondence represents a keypoint in an image which is detected and matched to the corresponding keypoint in the other image. Although there are a few erroneous matches, the majority are aligned with one homography, and RANSAC can find an accurate homography estimate from the matching keypoints between the two images.

In a texel image, with 3D points matched to corresponding pixels in the image, the homography is used to find point projections from one image into another image. The resulting homography between the two images is used as an initial estimate, and the true point projections are found using NCC. For a given 3D point inside the texel image j, a

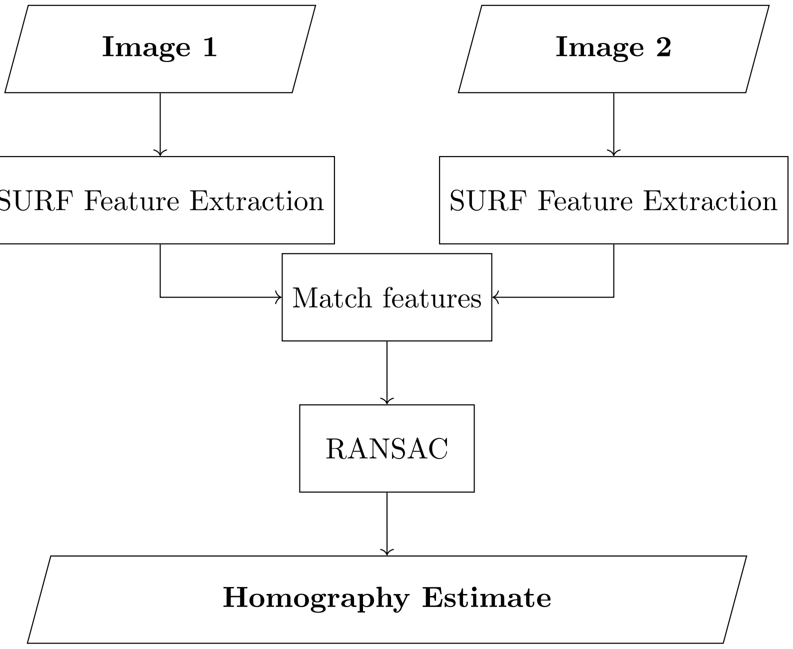


Fig. 2.7: Block diagram for homography estimation between two images using feature detection with SURF.

small patch of pixels is taken from the corresponding location on the image plane. This patch is then correlated to another patch on a different texel image k, using the homography from SURF as the initial estimate to reduce the search size. The location in k which has the highest correlation is used to refine the projection of the point from image j to image k.

Homography estimates work best between two images which have high overlap. Finding the homography between two images which are very similar and have minimal parallax results in more matched features and a more accurate estimate of the homography that fits the true model between the two images.

Correlation is used with the homography estimate as a starting point for the projections from a point into another image. NCC determines the similarity between two image patches. Given two images, Im_1 and Im_2, and pixels with column and row values in these images, $\mathbf{p}_1 = (c_1, r_1)$ and $\mathbf{p}_2 = (c_2, r_2)$, and μ_j the mean value of the patch j, the NCC match is given by the maximum γ found with the equation:

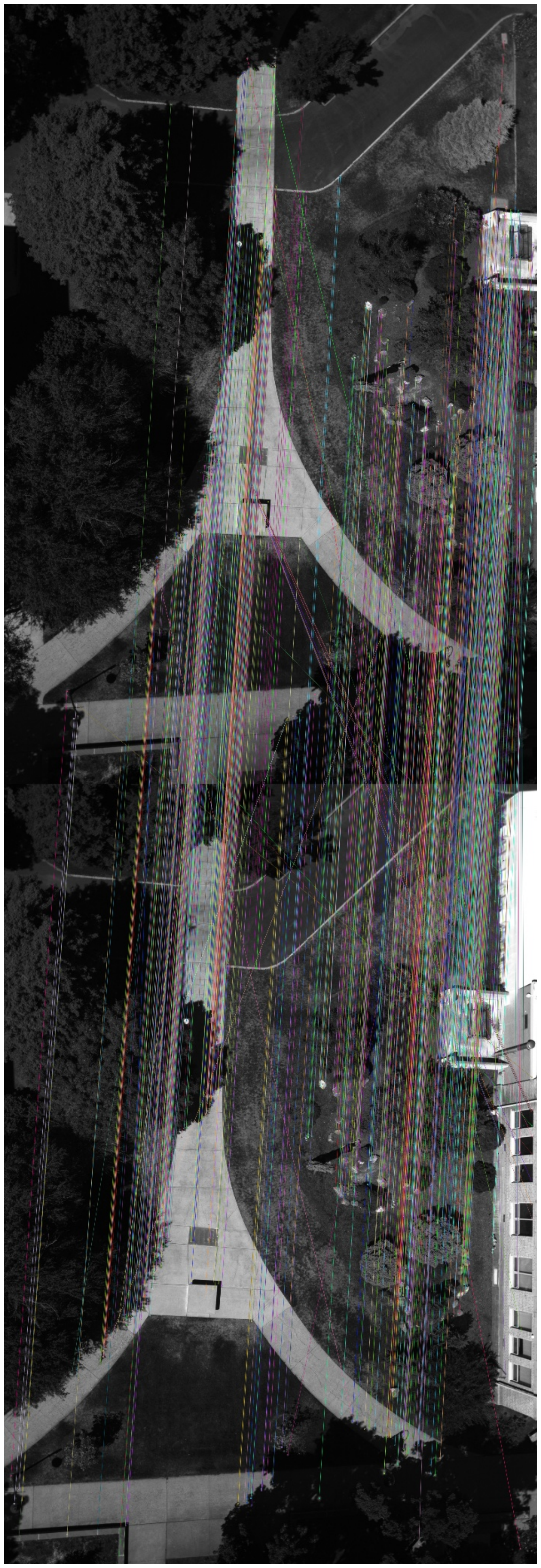


Fig. 2.8: Image correspondences matched between two optical images. A line is drawn between matching SURF features.

$$\gamma(\mathbf{p_1},\mathbf{p_2}) = \frac{\sum\limits_{r=-\frac{N}{2}}^{\frac{N}{2}} \sum\limits_{c=-\frac{M}{2}}^{\frac{M}{2}} (\mathrm{Im}_1(c_1+c,r_1+r)-\mu_1)(\mathrm{Im}_2(c_2+c,r_2+r)-\mu_2)}{\sqrt{\left(\sum\limits_{r=-\frac{N}{2}}^{\frac{N}{2}} \sum\limits_{c=-\frac{M}{2}}^{\frac{M}{2}} (\mathrm{Im}_1(c_1+c,r_1+r)-\mu_1)^2\right)\left(\sum\limits_{r=-\frac{N}{2}}^{\frac{N}{2}} \sum\limits_{c=-\frac{M}{2}}^{\frac{M}{2}} (\mathrm{Im}_2(c_2+c,r_2+r)-\mu_2)^2\right)}} \tag{2.9}$$

If the homography estimate is poor, the area searched with NCC may not include the section where the true point projection falls in. In some cases, such as with high parallax or if the terrain changed sufficiently in the time between when the two images were taken, the images in which it is searching may be dissimilar enough that the correlation cannot match to the correct point, and the point projection is not used. This means that the algorithm performs best on terrain with minimal height differences relative to the flight elevation.

2.4 Texel Registration and Projection Matrix

A single TDSM can be created from multiple texel images registered into a single reconstructed scene. The main steps used for the texel registration which is used in this research, modified and adapted from Bybee's original algorithm presented in Section 1.1, are:

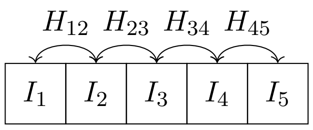
1. Estimate pairwise homographies between consecutive texel images.
2. Form the Projection matrix. This consists of the projection of each 3D point from the measured lidar data into every neighboring texel image which can see this point.
3. Apply bundle adjustment to optimize the 3D points and camera poses.
4. Find the appropriate texture and create the TDSM.

The projection matrix is formed with image processing techniques, using the homography computations and point projection equations outlined in Section 2.2. The pairwise homographies between neighboring images are found. Neighboring images are texel images which were acquired consecutively in time. To find the projection of a point into a non-neighboring image, the homographies from each intermediate neighboring pair are cascaded to obtain an estimate of the homography between the images.

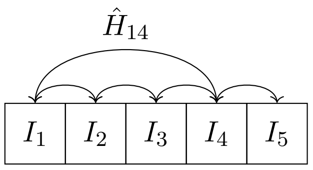
Figure 2.9 illustrates how the cascaded homographies are used to find the homography between two texel images which are not adjacent in time. I_i denotes the i^{th} texel image, H_ij denotes a homography which is directly calculated between two cascaded texel images, and $\hat{H}_{ij}$ denotes a homography estimate which is created by cascading adjacent homographies. Figure 2.9a shows which homographies are directly calculated. In Figure 2.9b, the homography estimate $\hat{H}_{14}$ is created by cascading H_{12}, H_{23}, and H_{34}.

Using the cascaded homographies allows the homographies to only be calculated between each adjacent pair; these are normally more accurate, as the rate at which images are captured means that two adjacent images are largely identical. Therefore, more feature points can be detected and extracted with SURF than from texel images which are not directly adjacent.

The projection matrix consists of the image projections for each 3D point onto all other texel images. For a point $\mathbf{b}_i$ in texel image j which is also seen by texel image k, the point can be projected into k to fill the entry in the matrix. The dimensions of the projection matrix is a block matrix of $m \times n$ blocks, with m points and n total texel images, and each block consisting of a 3-element vector, with the (x_{ij}, y_{ij}) values on the normalized image plane, and the range λ_i. As the range data is only obtained from the lidar sensor, the range is set as zero if the point is not a part of the texel image it is being projected into.



(a) Homographies found directly between texel images.



(b) Cascaded homographies to create a homography estimate between non-adjacent texel images.

Fig. 2.9: Example of cascaded homographies between texel images.

To illustrate this, an example is given with a projection matrix, with the texel images I_i and the pose of the j^{th} texel image $\mathbf{a}_j$, containing the points $\mathbf{b}_i$ such that $\mathbf{b}_1 \in I_1, \mathbf{b}_2, \mathbf{b}_3 \in I_2, \mathbf{b}_4 \in I_3$. The projection matrix for this example would consist of the normalized image plane points and ranges in the form:

$$\begin{bmatrix} [x_{11}, y_{11}, \lambda_{11}] & [x_{12}, y_{12}, 0] & [x_{13}, y_{13}, 0] \\ [x_{21}, y_{21}, 0] & [x_{22}, y_{22}, \lambda_{22}] & [x_{23}, y_{23}, 0] \\ [x_{31}, y_{31}, 0] & [x_{32}, y_{32}, \lambda_{32}] & [x_{33}, y_{33}, 0] \\ [x_{41}, y_{41}, 0] & [x_{42}, y_{42}, 0] & [x_{43}, y_{43}, \lambda_{43}] \end{bmatrix}. \tag{2.10}$$

If a point is not visible in a texel image, i.e. based on the homography estimate, no correlation could be found into that texel image, then the corresponding entry in the projection matrix is set to zero. The projection matrix is an important step which defines the model that the bundle adjustment attempts to optimize.

The projection matrix is used to compute the projection error and range error, which are used in bundle adjustment as the error to optimize inside the objective function. Projection error is the Euclidean distance between the estimated camera projection-range, $\hat{\mathbf{X}}_{ij} = [\hat{x}_{ij}, \hat{y}_{ij}, \hat{\lambda}_{ij}]^T$, and the measured projection-range, $\mathbf{X}_{ij} = [x_{ij}, y_{ij}, \lambda ij]^T$.

The error variances are σ_I^2, as the 2D projected point error, and σ_λ^2, as the range measurement error. These are defined from the texel camera hardware, and determined as part of the sensor calibration setup.

Given the $i^t h$ point $\mathbf{b}_i$ observed in the j^{th} texel image, the error ϵ_{ij} is calculated using:

$$\epsilon_{ij}^2 = \frac{1}{\sigma_I^2}(x_{ij} - \hat{x}_{ij}) + \frac{1}{\sigma_I^2}(y_{ij} - \hat{y}_{ij}) + \frac{1}{\sigma_\lambda^2}(\lambda_{ij} - \hat{\lambda}_{ij}). \tag{2.11}$$

Because the measured range λ_{ij} is only available when the owner of the i^{th} point is the j^{th} texel image (i.e., the point $\mathbf{b}_i$ was acquired in texel image I_j), the error is calculated differently by omitting the $\frac{1}{\sigma_\lambda^2}(\lambda_{ij} - \hat{\lambda}_{ij})$ term if the owner of the point $\mathbf{b}_i$ is not I_j. The error ϵ_{ij} is then calculated using:

$$\epsilon_{ij}^2 = \frac{1}{\sigma_I^2}(x_{ij} - \hat{x}_{ij}) + \frac{1}{\sigma_I^2}(y_{ij} - \hat{y}_{ij}). \tag{2.12}$$

Bundle adjustment is the non-linear optimization which refines the camera pose and position of 3D lidar points. This adjustment comes from minimizing the error in the estimated and measured point projections. With M total texel images and N total points inside the texel images, the error cost function to compute the squared norm of error $||\epsilon||^2$ combines the error equations 2.11 and 2.12, given as:

$$||\epsilon||^2 = \sum_{j=1}^{M}\left(\sum_{\substack{i=1\\ i\in I_j}} \frac{1}{\sigma_I^2}[(x_{ij}-\hat{x}_{ij})^2+(y_{ij}-\hat{y}_{ij})^2]+\sum_{\substack{i=1\\ i\in I_j}} \frac{1}{\sigma_\lambda^2}(\lambda_{ij}-\hat{\lambda}ij)^2+\sum_{\substack{i=1\\ i\notin I_j}} \frac{1}{\sigma_I^2}[(x_{ij}-\hat{x}_{ij})^2+(y_{ij}-\hat{y}_{ij})^2]\right). \tag{2.13}$$

It is not necessary for every entry in the projection matrix to be non-zero. If a measurement is missing, i.e. the point is not seen by that image and no projection into the image exists, the term is omitted from the sum.

The Levenberg-Marquardt algorithm was used for the bundle adjustment. The Levenberg-Marquardt algorithm is an iterative algorithm which combines the Gauss-Newton algorithm and gradient descent optimization, and can be used for multiple image bundle adjustment [23]. It is used because the Levenberg-Marquardt algorithm finds the maximum likelihood estimate based on the camera parameters and pose estimates recorded from the sensors at the time the texel image qas acquired. The size of the bundle adjustment can be adjusted based on the size of the data set and any processing or memory limitations, making it reasonable to run on any hardware.

The Levenberg-Marquardt algorithm adjusts the parameter vector P, which contains camera parameters $\mathbf{a}_j$ and the 3D points $\mathbf{b}_i$. The poses from the INS recorded at the time the texel images are captured are used the starting point for the optimization.

Let $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ be the function that takes the camera parameter and set of 3D points within each texel image as the input and computes the image projections and ranges on each image. In the optimization, given the initial estimate $P_0 \in \mathbb{R}^m$, the objective is to find the $\hat{P}$ with minimized $||\epsilon||^2$ satisfying $X = f(\hat{P}) - \epsilon$. The squared norm of error $||\epsilon||^2$ is given in 2.13.

The central normal equation for the Levenberg-Marquardt algorithm is given as

$$(\mathbf{J}^T\mathbf{\Sigma^{-1}}\mathbf{J} + \rho \operatorname{diag}(\mathbf{J}^T\mathbf{\Sigma^{-1}}\mathbf{J}))\delta\mathbf{x} = \mathbf{J}^T\mathbf{\Sigma}^{-1}\epsilon. \tag{2.14}$$

with the variables in 2.14 defined

$\mathbf{J} = \frac{\partial f}{\partial \mathbf{P}}$ as the Jacobian,

$\mathbf{\Sigma}$ as the covariance matrix of the measurement,

ρ as the damping parameter for the Levenberg-Marquardt algorithm,

$\delta\mathbf{x}$ as the parameter update vector.

The parameter vector $\mathbf{P}$ is created with $P = [\mathbf{a}^T, \mathbf{b}^T]^T$ so that the Jacobian has the form $\mathbf{J} = [A|B]$, with $A = \frac{\partial \hat{X}}{\partial \mathbf{a}}$, $B = \frac{\partial \hat{X}}{\partial \mathbf{b}}$. The table for the Jacobian matrix in bundle adjustment, containing the points $\mathbf{b}_i$ such that $\mathbf{b}_1 \in I_1, \mathbf{b}_2, \mathbf{b}_3 \in I_2, \mathbf{b}_4 \in I_3$ is:

$$\mathbf{J} = \frac{\partial \hat{X}}{\partial P} = \begin{array}{c|ccccccc}
 & a_1 & a_2 & a_3 & b_1 & b_2 & b_3 & b_4 \\
\hline
\boldsymbol{X_{11}} & A_{11} & 0 & 0 & B_{11} & 0 & 0 & 0 \\
\boldsymbol{X_{12}} & 0 & A_{12} & 0 & B_{12} & 0 & 0 & 0 \\
\boldsymbol{X_{13}} & 0 & 0 & A_{13} & B_{13} & 0 & 0 & 0 \\
\boldsymbol{X_{21}} & A_{11} & 0 & 0 & 0 & B_{21} & 0 & 0 \\
\boldsymbol{X_{22}} & 0 & A_{12} & 0 & 0 & B_{22} & 0 & 0 \\
\boldsymbol{X_{23}} & 0 & 0 & A_{13} & 0 & B_{23} & 0 & 0 \\
\boldsymbol{X_{31}} & A_{11} & 0 & 0 & 0 & 0 & B_{31} & 0 \\
\boldsymbol{X_{32}} & 0 & A_{12} & 0 & 0 & 0 & B_{32} & 0 \\
\boldsymbol{X_{33}} & 0 & 0 & A_{13} & 0 & 0 & B_{33} & 0 \\
\boldsymbol{X_{41}} & A_{11} & 0 & 0 & 0 & 0 & 0 & B_{41} \\
\boldsymbol{X_{42}} & 0 & A_{12} & 0 & 0 & 0 & 0 & B_{42} \\
\boldsymbol{X_{43}} & 0 & 0 & A_{13} & 0 & 0 & 0 & B_{43}
\end{array} \tag{2.15}$$

The Jacobian is a sparse matrix when a large number of images are used in a single bundle adjustment, as most of the entries in the matrix in (2.15) are zero.

The update vector is then $\delta\mathbf{x} = [\delta\mathbf{a}|\delta\mathbf{b}]$. The normal equation is

$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix} \begin{pmatrix} \delta\mathbf{a} \\ \delta\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{A}} \\ \epsilon_{\mathbf{B}} \end{pmatrix}. \tag{2.16}$$

The matrix entries are defined as

$$\mathbf{U} = A^T\mathbf{\Sigma}^{-1}A,$$

$$\mathbf{W} = A^T\mathbf{\Sigma}^{-1}B,$$

$$\mathbf{V} = B^T\mathbf{\Sigma}^{-1}B,$$

$$\epsilon_{\mathbf{A}} = A^T\mathbf{\Sigma}^{-1}\epsilon,$$

$$\epsilon_{\mathbf{B}} = B^T\mathbf{\Sigma}^{-1}\epsilon.$$

In a large bundle adjustment, the matrix on the left-hand side has a sparse structure from $\delta\mathbf{a}$ and $\delta\mathbf{b}$. A sparse matrix solver can be applied to solve this problem. After finding the new error, the increment is saved if the squared norm error from 2.13 is reduced, and the damping parameter, ρ, is adjusted accordingly. The parameter ρ is reduced by a factor of 10 if the error was reduced, or increased by a factor of 10 if the error was increased. This aids the Levenberg-Marquardt algorithm in converging to the final solution by transitioning between Gauss-Newton and gradient descent algorithms as it converges to the optimized solution.

2.5 Streaming Bundle Adjustment

The streaming bundle adjustment algorithm, first developed and introduced by Khatiwada, [1] reduces the size of the bundle adjustment in texel image registration, from a sparse matrix bundle adjustment to an iterative sliding window approach. A set of texel images arranged consecutively is given as the input to the algorithm. A sliding window of texel images is taken for a single bundle adjustment, the parameters for this window are optimized, and then the sliding window is moved to use the next texel images for another bundle adjustment. This process repeats until every texel image within the set has been optimized.

In the streaming bundle adjustment algorithm, the parameter *look length*, denoted with $\mathcal{L}$, is used to set how many texel images are included in a bundle adjustment. $\mathcal{L}$ defines the effect of one texel image onto its neighboring texel images. A texel image j has no point projections outside of a window of $\mathcal{L}$ texel images forward and backward in time. The look parameter should be set according to the amount of overlap one texel image has to its neighbor; ideally, each texel image should have projection points onto neighboring texel images within $\mathcal{L}$ forward and backward in time, and no point projections into texel images after or before this window. When creating the projection matrix, the point projections are found only within $\mathcal{L}$ texel images before and after the current texel image.

The size of the sliding window for a single iteration inside the streaming algorithm is set at $3\mathcal{L}$ texel images. Each iteration saves the oldest $\mathcal{L}$ texel images with the optimized points and poses inside the projection matrix, keeps the data according to $2\mathcal{L}$ in memory, and loads in the next $\mathcal{L}$ texel images for optimization.

Limiting the texel images used in bundle adjustment reduces the computations required. The use of a sliding window to restrict the maximum size of each bundle adjustment helps to ensure that running the algorithm is feasible on low-cost hardware, while minimizing the amount of memory and computational time required.

In this thesis, texel images captured consecutively in time are referred to as time-adjacent texel images, or sliding window texel images. Other texel images not in this sliding window, but which contain overlap with the sliding window texel images, are referred to as overlapping texel images.

2.6 Convex Hulls

For a set $S \subseteq \mathbb{R}^n$, the smallest convex set containing S is defined as the convex hull of S. The convex hull of S consists of all the convex combinations of the elements in S.

Convex hulls are used in this research to estimate the area of terrain which a given texel image covers. The convex hull is calculated using the (X, Y) coordinates from the 3D lidar points in the world frame, to create a set of points defined in $\mathbb{R}^2$. This simplifies the 3D point cloud from the lidar data to a set of 2D points. The vertices of the convex polygon which define the convex hull are calculated.

The convex hull is obtained using an algorithm known as Graham's scan. This algorithm was chosen for its efficiency, with a time complexity of $\mathcal{O}(n \log n)$ to calculate the convex hull for n points. For a starting X, Y point, the remaining points in the set are sorted in descending order by the angle formed with the starting point. For each of the points in this list, it is determined whether moving to this point from the previous two points requires a clockwise or counter-clockwise rotation. The clockwise or counter-clockwise rotation is determined by calculating the z-coordinate of the cross product, considering the point P_2 in the list and the preceding two points, P_1 and P_0, with the cross product taken between the vectors $\overrightarrow{P_0P_1}$ and $\overrightarrow{P_0P_2}$. The equation for the z-coordinate is:

$$z = P_{0x}(P_{1y} - P_{2y}) + P_{1x}(P_{2y} - P_{0y}) + P_{2x}(P_{0x} - P_{1y}). \tag{2.17}$$

If the z-coordinate is positive, then it is a counter-clockwise rotation and the points P_1, P_0 are kept; otherwise, it is a clockwise rotation between the vectors, and the point P_1 is inside the convex hull, and can be discarded as it is not a vertex for the convex hull.

This continues until the original starting point is returned to, and the resulting set of points are the vertices in the convex polygon defining the convex hull. This algorithm has $\mathcal{O}(n \log n)$ complexity for n points.

Figure 2.10 shows an example of a convex hull, with (X, Y) points obtained from a texel image. The convex hull represents the location in the world coordinate terrain, with the axes in meters. The locations on the X- and Y- axes are referenced to the first texel image acquired in the data set, in world coordinates; in the X direction, this example starts about thirty meters east from where the flight first started acquiring data.

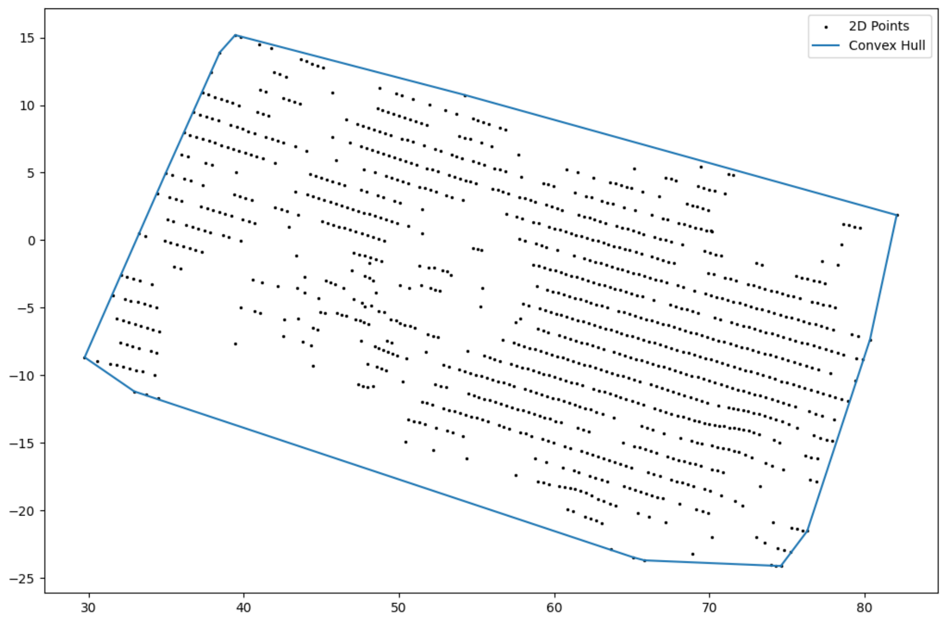


Fig. 2.10: Convex hull generated from data points in a texel image.

2.7 Texel image data set

To acquire real-world data, the texel camera constructed at the Center for Advanced Imaging Ladar (CAIL) was mounted to a sUAV and was flown over a desired area of terrain to capture texel images. The texel camera on the sUAV has an optical camera, lidar sensor, and INS to save the camera pose and reference the acquired 3D points to the world frame.

The field of view of the lidar is 15.8 degrees in the vertical direction, and configurable in the horizontal direction between 30 and 40 degrees. The lidar can rotate 360 degrees at a rate of 10 Hz.

The digital camera has a maximum resolution of 4240 by 2832 pixels. Because the field of view for the digital camera is greater than the field of view for the lidar sensor, the digital image is cropped and the region of interest is adjusted so that both the optical camera and lidar sensor have the same field of view.

The texel camera is designed to use inexpensive parts, to allow for a wide variety of applications it can be used for. Because of this, the INS in the texel camera attached to the sUAV contains noticeable errors that must be corrected for registration. The pose provided by the INS gives the position within 3 meters and the attitude to within 0.3 degrees in roll, pitch, and yaw. The coarse data provided by the INS motivates the utilization of image processing to correct for errors when registering images.

CHAPTER 3

Methods

The focus of this research is to adapt the streaming bundle adjustment algorithm to account for texel images not adjacent in time which have overlap with the texel images contained in the streaming window. The objective of this is to improve the accuracy of the reconstructed TDSM by searching the data set outside of the streaming window to check for additional texel images which can reduce the error to the true terrain after bundle adjustment.

To do this, the data set is searched to find other texel images which overlap with the same area of terrain the sliding window is optimizing, and then these overlapping texel images are provided to the streaming algorithm to augment the bundle adjustment. To ensure that the poses can be corrected on low-cost hardware, optimizations are made to this algorithm to reduce the computational complexity and memory usage.

In the algorithm developed in this research, the desired texel images for TDSM reconstruction are provided as input to the program. The input texel images are arranged consecutively in time, and may be a small subset of the available flight data. The input texel images are sorted into time-adjacent sliding windows which correspond to a single bundle adjustment. For each of these sliding windows, the entire data set is searched for overlapping texel images. These non-adjacent texel images are added to the bundle adjustment to improve the resulting TDSM.

This means that before any bundle adjustment is done, an instruction set is created that informs the bundle adjustment which texel images to use for each bundle adjustment. The instruction set contains both the sliding window texel images to optimize, and the overlapping texel images which should be included in the optimization. These instructions are then read in by the program, bundle adjustment is performed, and the resulting reconstructed TDSM is saved.

The main steps developed are:

1. Load each texel image and calculate the convex hull of its point cloud, projected onto the (X, Y) plane. These convex hulls are used to estimate overlap between texel images which are not adjacent in time.

2. With the overlap estimates, augment the time-adjacent sliding windows to account for this overlap.

3. Perform bundle adjustment using the Levenberg-Marquardt algorithm with the augmented sliding windows.

In this research, the assumption was made that all texel images from the flight data are available at the time of TDSM registration. This is used so that, for a given texel image, other overlapping texel images which occur much later in the flight are able to be used. This allows for the pose to be corrected using the entire available flight data. No assumption is made about the flight pattern or that a given texel image has any overlap with other texel images not adjacent in time.

A limitation inherent from this assumption is that the registration cannot be done in real-time as the flight data is being captured, because texel images captured later in time are used in this algorithm; however, the high computational costs of bundle adjustment in correcting the errors already prohibit real-time registration, so it was decided this is an acceptable assumption to make.

3.1 Estimating Overlap

A method of overlap detection for texel images was developed. This method estimates the area of terrain captured by a given texel image, in the world coordinate frame, and searches for other texel images in the flight data set which also capture this area of terrain. Before bundle adjustment occurs, each texel image from the full flight data is loaded and the 3D point cloud data is processed.

Each texel image contains 3D points in (X, Y, Z) coordinates in the camera coordinate system. Using the camera pose obtained from the texel camera's INS, these points can be transformed into the world coordinate frame, in East, North, Up (ENU) coordinates. The equations to perform this projection are outlined in Section 2.2.

The convex hull data for an entire flight is saved as a set of vertices in X, Y coordinates, so it is quick to run computations on this set of data. Because it is used to estimate overlap between two terrain scans, creating 2D convex hulls from 3D data points is sufficient for this application. To calculate the convex hull for a texel image, the set of 3D points from a single texel image are transformed into world coordinates, and projected onto the X, Y plane. The convex hull of this set of (X, Y) points is calculated for each image and saved as a set of vertices defining the convex polygon as described in Section 2.6.

This convex hull created from a single texel image can be thought of as a rough estimate of the terrain that the texel image covers, in world space coordinates; all lidar points are contained inside this convex hull, and the rotational lidar sensor is calibrated to have the same field of view as the optical camera. As this data still contains the uncorrected error inside the camera pose, it is only used as an initial estimate.

Another potential method for estimating overlap would be to project the field of view of the LiDAR sensor onto the terrain based on the estimated range of the captured 3D points from the camera. However, calculating the convex hulls with the points is computationally efficient and provides an accurate estimate, so it was chosen as the method to implement for this research.

The percentage of overlap that a texel image i has with another texel image j is calculated from their convex hulls. Let $\mathcal{C}_i$ be the convex hull of the i^{th} texel image, and $\mathcal{C}_{ij}$ be the polygon representing the intersection of $\mathcal{C}_i$ and $\mathcal{C}_j$. The overlap value ρ_{ij} is defined as:

$$\rho_{ij} = \frac{\mathbf{area}\mathcal{C}_i}{\mathbf{area}\mathcal{C}_{ij}}. \tag{3.1}$$

An illustration of convex hulls with a set of 300 texel images, taken from the data that this research was tested on, is shown in Figure 3.1. Each polygon represents the convex hull generated from a single texel image. In the flight pattern, the drone captured texel images starting from the top-left, then progressed down the left column and up the right column, ending at the top-right texel image. This represents two legs of the flight. Overlap between non-consecutive texel images can be seen in this figure, such as in the center where the polygons overlap.

In Figure 3.2, a portion of the convex hulls from the same set are scaled and aligned to the flight path from which they were captured. The convex hulls are shown in blue for visual clarity. This shows how the convex hulls correspond to the area of terrain the texel images capture, and how texel images which are not adjacent in time contain overlap with each other.

In the example illustrated, the two legs happen to be acquired after each other from the flight pattern; this is not necessary for overlap to be detected. No assumption about the ordering of the texel images is made when determining overlap, so overlap can be found within any texel image in the data set.

After the convex hulls are obtained, they are used to create an instruction set for augmenting the sliding windows used in bundle adjustment.

This step is primarily influenced by two parameters which can be modified: ρ and $\mathcal{M}$. ρ is the minimum percent of overlap the texel image must have with the sliding window polygon in order to be included in the augmented bundle adjustment. $\mathcal{M}$ is defined as the maximum number of overlapping texel images added to each bundle adjustment.

The minimum percent overlap parameter ρ is to increase the chance that, for each overlapping texel image, accurate point projections to this can be found. If the minimum overlap is too small, fewer feature correlations will be found, and it is less likely that an accurate homography between them will be found. The maximum number of overlapping texel images to save is used so that, if there is an irregular flight pattern which contains a large amount of frames overlapping, only the frames which contain the most overlap will be used. A higher number for the maximum frames parameter may result in redundant frames being added, which increase computational time while not improving the result of the final registration. $\mathcal{M}$ places a limit on the additional computation for an augmented

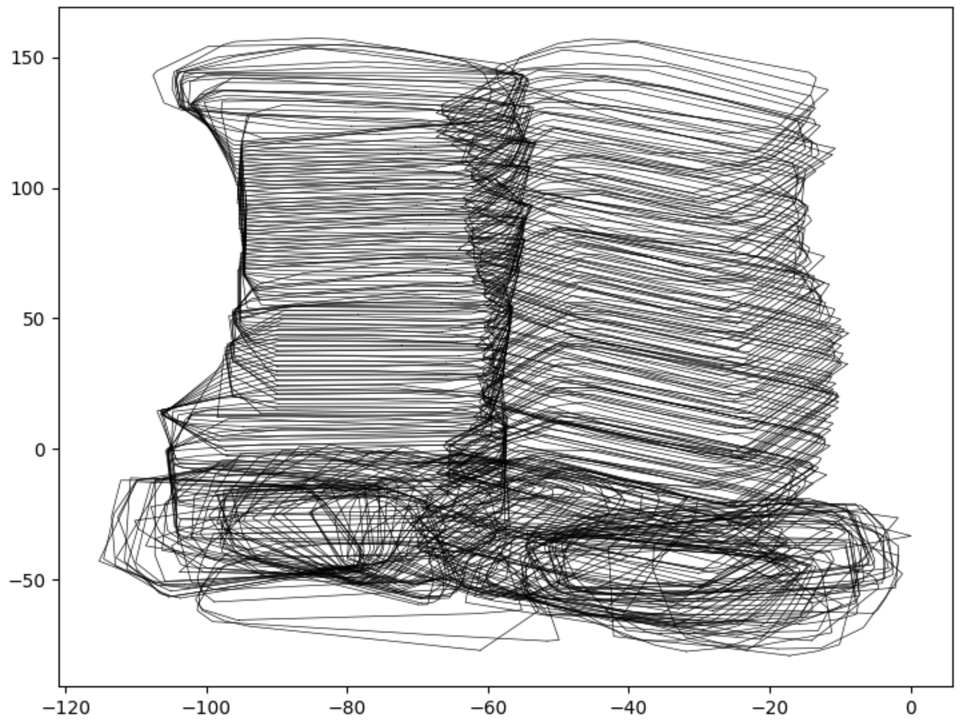


Fig. 3.1: Texel image convex hulls. These convex hulls are taken from from texel images acquired during adjacent legs of a survey flight.

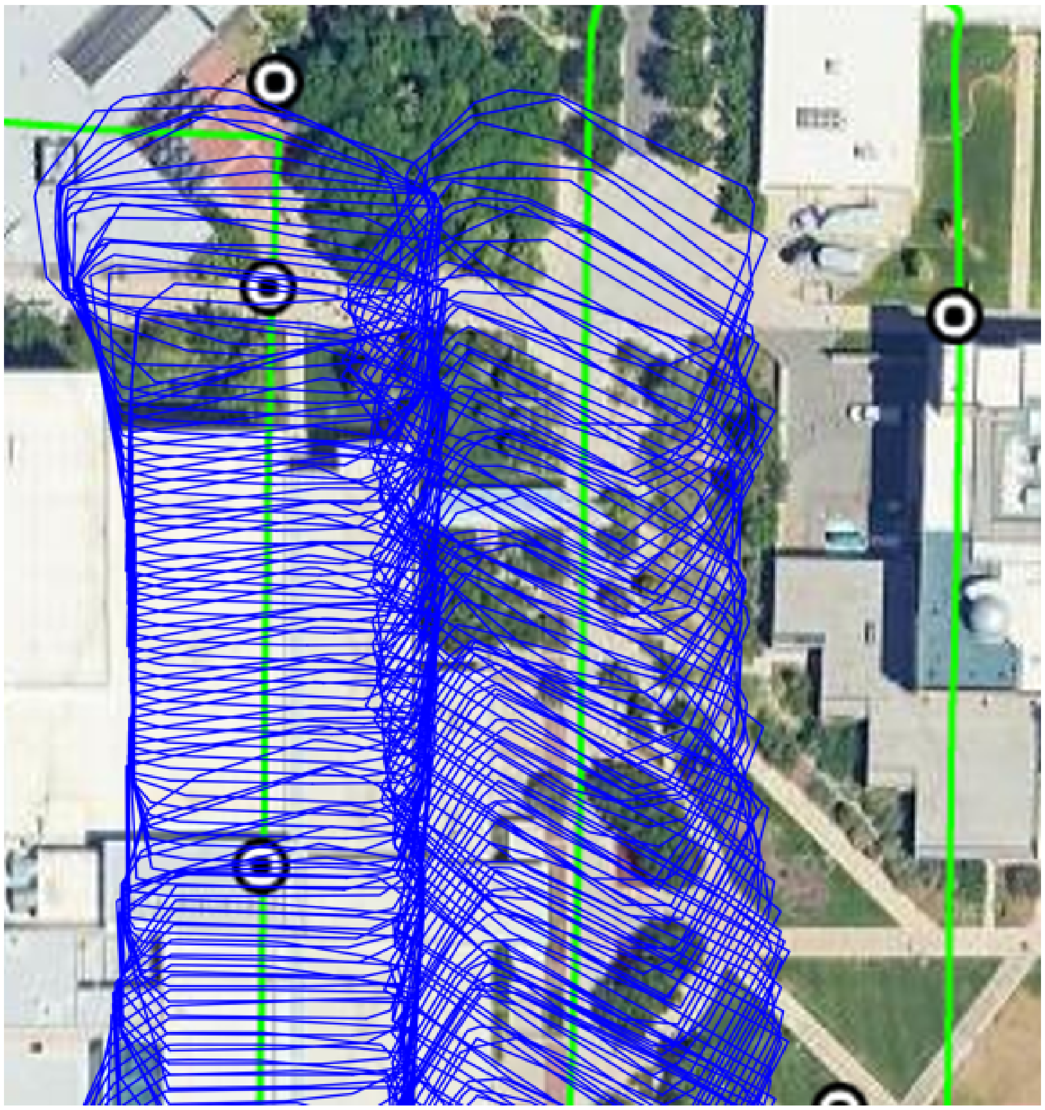


Fig. 3.2: Convex hulls generated from texel images, overlaid on flight path.

bundle adjustment, when compared with a non-augmented bundle adjustment.

The instruction set for which overlapping frames to use is calculated from an input defining a consecutive range of texel images to register together in the final TDSM. Inside this step, no bundle adjustment or projection computations are done; the program is only using the convex hull data to generate a set of vectors defining the indices of the images to read in for each sliding window bundle adjustment.

Given the input range of consecutive texel images and the defined look parameter $\mathcal{L}$, this range is separated into the sliding windows used in the streaming bundle adjustment. This creates a matrix with m rows, where m is the number of bundle adjustments performed, and $3\mathcal{L}$ columns. The total number of bundle adjustments is defined by the look parameter and total range of texel images. A single bundle adjustment is referred to as a single optimization. This optimization consists of bundle adjustment on $3\mathcal{L}$ texel images, with $\mathcal{L}$ texel images optimized and saved after the Levenberg-Marquardt algorithm. If there are j texel images in this range and $\mathcal{L}$ is a factor of j, then the total number of bundle adjustment optimizations is:

$$m = \frac{(j - 3\mathcal{L})}{\mathcal{L}} \tag{3.2}$$

If $\mathcal{L}$ is not a factor of j, the range must be truncated to remove the last $j \bmod \mathcal{L}$ images to give $j - (j \bmod \mathcal{L})$, texel images, so the total number of optimizations is:

$$m = \left\lfloor \frac{(j - 3\mathcal{L})}{\mathcal{L}} \right\rfloor \tag{3.3}$$

As an illustration, if the texel image indices range is from t_1 to t_j, with t_i as the index of a single texel image, the matrix defined as $\mathcal{S}$ consists of the sliding window instruction set.

$$\mathcal{S} = \begin{bmatrix} \mathbf{s}_0 \\ \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_m \end{bmatrix} = \begin{bmatrix} t_1 & t_2 & \cdots & t_{3\mathcal{L}} \\ t_{\mathcal{L}} & t_{\mathcal{L}+1} & \cdots & t_{4\mathcal{L}} \\ \vdots & \vdots & \ddots & \vdots \\ t_{j-3\mathcal{L}} & t_{j+1-3\mathcal{L}} & \cdots & t_j \end{bmatrix} \tag{3.4}$$

Each row of $\mathcal{S}$ consists of the indices of the consecutive images to use in a single optimization of the bundle adjustment. We have:

$\mathcal{S}$ = the $m \times 3\mathcal{L}$ matrix of indices.

$\mathbf{s}_i$ = the set of indices of texel images used in the sliding window for the i^{th} optimization of bundle adjustment.

$\mathcal{C}_u$ = the union of the convex hulls inside $\mathbf{s}_i$, described as a set of vertices defining the convex polygon.

Let $\mathcal{C}_i$ be the polygon for the i_{th} convex hull, and $\mathcal{C}_u$ be the convex hull for the sliding window polygon, defined as:

$$\mathcal{C}_u = \bigcup_{i=1}^{3\mathcal{L}} \mathcal{C}_i \tag{3.5}$$

.

with $i = 1, \ldots, 3\mathcal{L}$ as the set of indices for the texel images in the sliding window.

After the index matrix is found, the overlap with non-consecutive images is calculated for each bundle adjustment optimization $\mathbf{s}_i$. The parameters ρ and $\mathcal{M}$ are used to find which overlapping frames are used in augmenting the sliding window. The process for this can be defined with the steps:

1. Determine $\mathcal{C}_u$ as the union of convex hulls for the texel images inside $\mathbf{s}_i$.

2. For each texel image in the flight data not in $\mathbf{s}_i$, calculate the intersection of its convex hull with $\mathcal{C}_u$. If the overlapping percentage of this intersection is less than ρ, there is not sufficient overlap for this algorithm and the texel image can be discarded.

3. After each intersection is calculated, take the list of all texel images with overlap and keep the $\mathcal{M}$ images with the highest overlap.

When implementing this algorithm to find the intersection between two polygons, the Shapely library was used in a Python script. [24]

To illustrate the concept of detecting overlap with sliding window sections, a section of texel images corresponding to the upper part of Figure 3.1 was taken, shown in Figure 3.3. There are 21 convex hulls each on the left and right side, which matches to the length of a single sliding window with look parameter $\mathcal{L} = 7$. In this example, consider the left leg to be the texel convex hulls corresponding to the texel images inside a single sliding window optimization, and the right leg to the non-adjacent texel images to be considered for overlap.

The blue polygon in Figure 3.4 shows the union of convex hulls for the sliding window, C_u. This polygon is the area which is analyzed for overlap to add additional swaths to. Every individual convex hull, as the black polygons in this example, are checked for overlap with the sliding window.

The area which overlaps with the sliding window is shown in Figure 3.5, filled in with cyan. This area is used to calculate the overlap percent.

The area which overlaps is then found from the intersection of the overlap image convex hull and the sliding window polygon, $C_o \bigcap C_u$. The overlap percent is defined as ρ_o, for the overlap that the o^{th} non-adjacent texel image C_o has with the sliding window polygon C_u, calculated with:

$$\rho_o = \frac{\textbf{area } C_o}{\textbf{area } C_o \bigcap C_u} \tag{3.6}$$

From Equation (3.6), the range of ρ_o is $0 \leq \rho_o \leq 1$, so the overlap percentage can be defined as $100 \times (\rho_o)$.

In the example given, the ρ_o value for each of the overlapping convex hulls C_o is calculated. The $\mathcal{M}$ texel images with the highest ρ are saved. With the parameter set $\mathcal{M} = 5$, the five most overlapping texel images are taken, and shown in Figure 3.6.

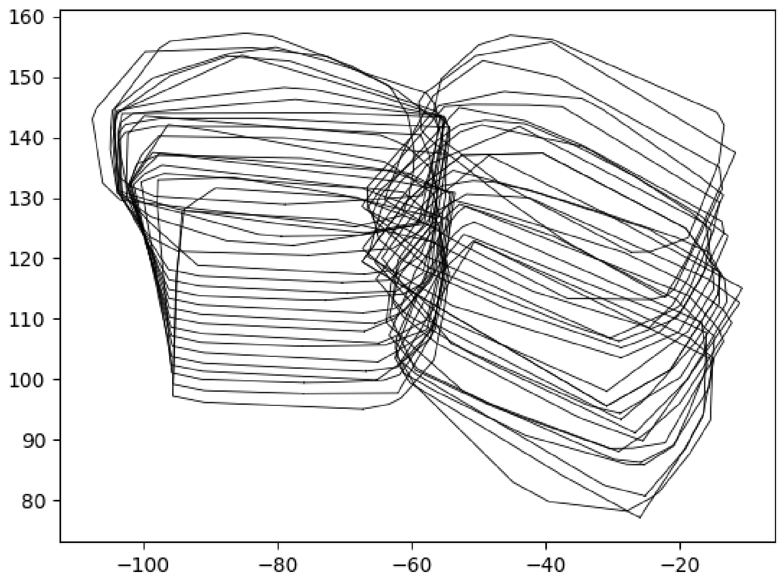


Fig. 3.3: A section of convex hulls generated from texel images.

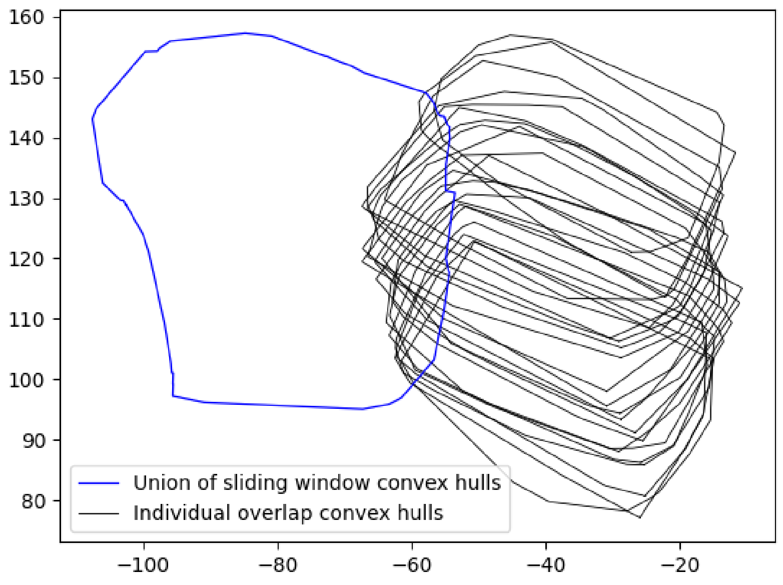


Fig. 3.4: A polygon representing the union of sliding window convex hulls, along with the overlap convex hulls to search for overlap.

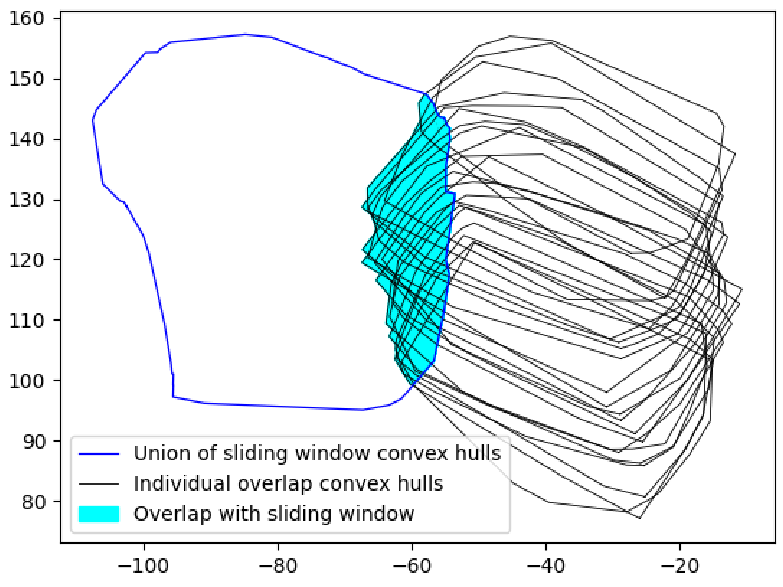


Fig. 3.5: Overlapping area within convex hulls.

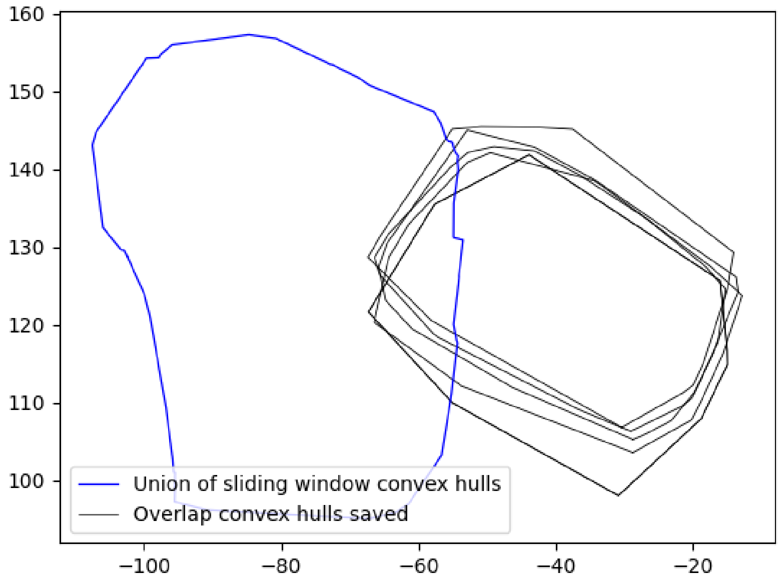


Fig. 3.6: Chosen convex hulls to augment the sliding window bundle adjustment.

The overlapping outlined convex hulls in Figure 3.6 correspond to the overlapping texel images that are used in the optimization, so the indices for these texel images are saved.

The final step for a single sliding window is to determine which single texel image within the sliding window each convex hull has the highest overlap with. This data is used in finding the homography estimates. For each overlapping convex hull, the texel image within the sliding window which it has the highest overlap with is saved; an illustration is in Figure 3.7, and the visualization of the overlap and the texel image with the highest intersection is presented in Figure 3.8 and Figure 3.9.

The most overlapping convex hull within the sliding window, shown as the black outlined polygon in 3.9, is required because the homography from the overlap texel image to the sliding window texel image is needed for the correlation estimates. Because the homography for correlation estimates are found from only two individual texel images, finding the polygon inside the sliding window which has the highest overlap helps the SURF feature detection find more matching keypoints, thus increasing the likelihood of an accurate homography being found. A higher overlap also gives more point projections within the overlapping area, helping the bundle adjustment accurately correct for error.

When calculating the overlap for a sliding window, depending on the flight pattern it is possible that very few or no overlapping texel images will be found. When this occurs, the bundle adjustment is done with only the time-consecutive texel images in the sliding window. Additionally, the overlap algorithm works when there are not enough overlapping texel images which meet the restriction of ρ; any number of overlapping swaths $\leq \mathcal{M}$ can be used.

Let $\mathbf{r}_i = [r_{i1}, r_{i2}, \cdots, r_{i\mathcal{M}}]$ be the vector of overlapping texel images which are found from this algorithm, with the $m \times \mathcal{M}$ matrix $\mathcal{R} = [\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_m]^T$. As part of r_{ij}, the sliding window texel image which has the greatest overlap with the overlapping texel image is also calculated and saved. The matrix $\mathcal{R}$ is appended to the full instruction set to get the augmented index matrix $\hat{\mathcal{S}} = [\mathcal{S}|\mathcal{R}]$ as the combination of these two matrices.

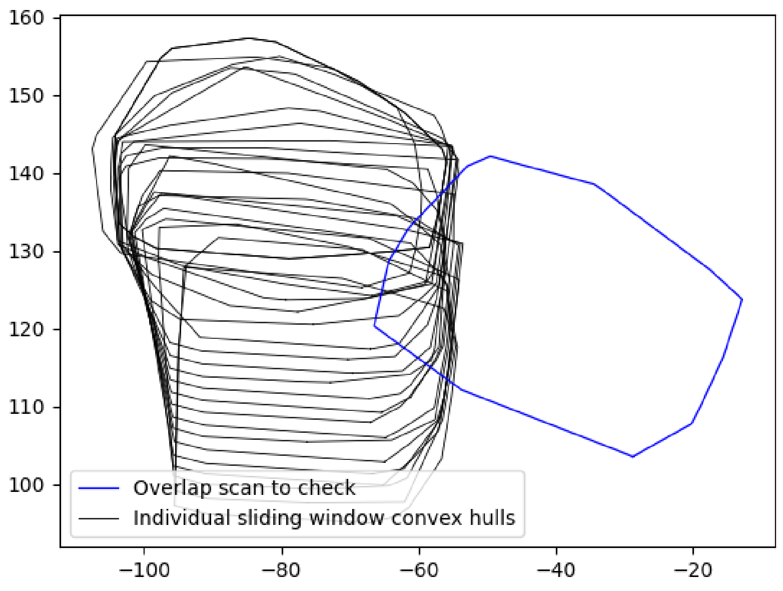


Fig. 3.7: Convex hulls within the sliding window to search for highest overlap.

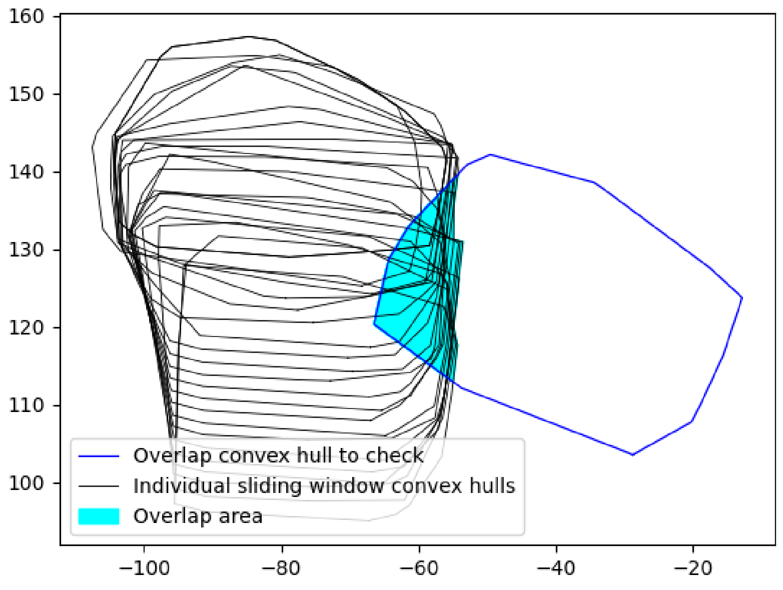


Fig. 3.8: Illustration of overlap area searched to find the highest overlapping area.

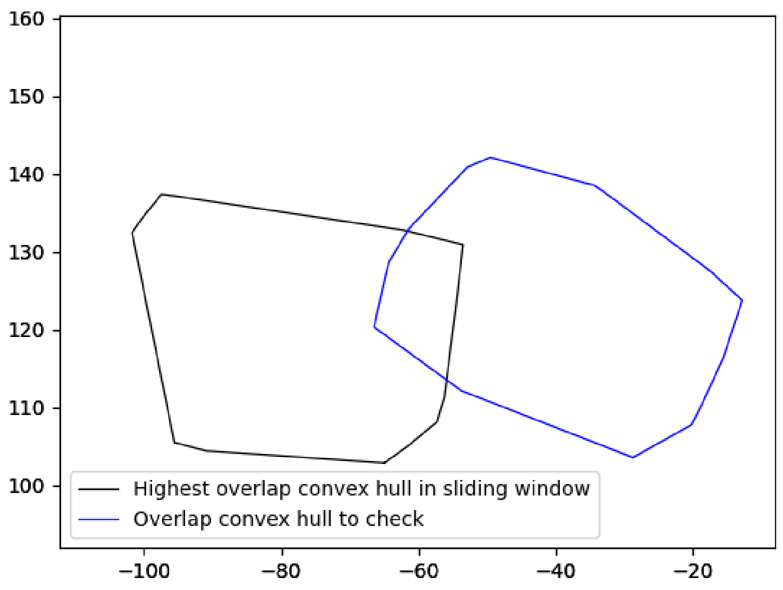


Fig. 3.9: The convex hull within the sliding window which has the highest overlap.

$$\hat{\mathcal{S}} = \begin{bmatrix} \hat{\mathbf{s}}_1 \\ \hat{\mathbf{s}}_2 \\ \vdots \\ \hat{\mathbf{s}}_m \end{bmatrix} = \begin{bmatrix} t_1 & t_2 & \cdots & t_{3\mathcal{L}} & r_{11} & \cdots & r_{1\mathcal{M}} \\ t_{\mathcal{L}} & t_{\mathcal{L}+1} & \cdots & t_{4\mathcal{L}} & r_{21} & \cdots & r_{2\mathcal{M}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ t_{j-3\mathcal{L}} & t_{j+1-3\mathcal{L}} & \cdots & t_j & r_{m1} & \cdots & r_{m\mathcal{M}} \end{bmatrix} \tag{3.7}$$

In the case where $\mathcal{R}$ is an empty matrix, i.e. there are no overlapping images appended to the matrix, the bundle adjustment is identical to the non-augmented streaming bundle adjustment.

The matrix $\hat{\mathcal{S}}$ contains the instructions for the augmented bundle adjustment. For each optimization i of the bundle adjustment algorithm, the texel images corresponding to the indices in $\hat{\mathbf{s}}_i$ are loaded in. For the next optimization $i+1$, the texel images in $\hat{\mathbf{s}}_i$ which are also present in $\hat{\mathbf{s}}_{i+1}$ are retained in memory. The texel images retained in memory correspond to the overlapping texel images which are the same, and the $2\mathcal{L}$ texel images inside the sliding window kept for the next bundle adjustment in the streaming algorithm's sliding window.

3.2 Augmented Bundle Adjustment

The augmented bundle adjustment uses the row vectors $\hat{\mathbf{s}}_i \in \hat{\mathcal{S}}$ as the indices for the set of texel images to load in.

The projection matrix must be found for every texel image loaded in. The projection matrix with the appended entries from the overlapping texel images is referred to in this work as the augmented projection matrix. For the $3\mathcal{L}$ texel images which are within the sliding window, calculating the projection matrix is the same as the streaming bundle adjustment. This uses the set of pairwise homographies between two texel images cascaded to obtain an estimate of the homography without directly calculating it, and then using normalized cross correlation to refine the location of the point estimate.

The points inside the overlapping texel images are appended to the projection matrix. To find the projection of a point within the sliding window texel images into the overlapping texel images, or the overlapping into the sliding window, the same set of consecutive pairwise homographies cannot be solely used. Instead, the homography is calculated between the overlap image and the sliding window texel image it has the highest overlap with, which was found for each overlapping texel image in the overlap estimation step. The index of this sliding window texel image is provided as part of $\mathbf{r}_i$.

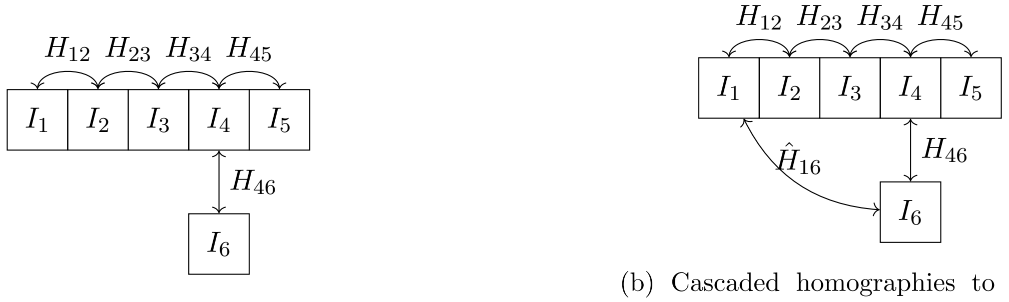
Compared to finding projections between texel images within the sliding window, finding the projections between sliding window and overlap texel images only differs in how the homography estimates are initially found. Other steps, such as the normalized cross correlation and transformation into the normalized image planes, are identical. Adding overlap point projections to the matrix requires the algorithm to accommodate for two additional cases: point projections between a sliding window image and an overlap image, and point projections between two overlap images.

For finding point projections between overlap swaths and the sliding window, the cascaded pairwise homographies are used. Let I_o denote the overlapping texel image, and I_j denotes the texel image within the sliding window which has the highest overlap with I_o. The first step in finding point projections is finding the homography directly between I_j

and I_o, using SURF and RANSAC in an identical process to finding homographies within the sliding window. This assumes that I_j and I_o have enough overlap for SURF to find enough keypoints that RANSAC is able to find an accurate homography. Inaccurate homographies may have a significant negative impact on the projection matrix, so at this step, the homographies are automatically checked to discard homographies that are likely to be inaccurate. If SURF cannot find enough keypoints to meet a parameter (chosen by the user), either because the overlapping portion of the images is too small or the overlapping image is too homogenous and thus does not have enough unique features, then it is assumed a homography fitting the true model will not be found, and the overlapping texel image is discarded entirely. Another check to verify that the overlap homography estimate is accurate is using the RANSAC confidence parameter. If it is too low, i.e. RANSAC is not able to find an accurate consensus which fits the majority of the points in the model, then an accurate homography is not found and the overlapping texel image is discarded and not used.

The homography between I_o and I_j is found directly. For the point projections from I_o into other texel images in the sliding window and vice versa, the consecutive pairwise homographies already found for the sliding window are cascaded with this homgraphy to find the estimate. Within the pairwise homography, $\mathcal{L}$ defines the maximum number of concatenated homographies. From the definition of the look parameter $\mathcal{L}$, the points in a given texel image j are not visible after $\mathcal{L}$ images forward and backward in time; so if an overlapping texel image has the highest overlap with image inside the sliding window i, the overlapping texel image also does not have point projections in the $j - \mathcal{L}$ or $j + \mathcal{L}$ consecutive texel images within the sliding window.

An example is shown in Figure 2.9, with I_i denoting the i^{th} texel image, and I_6 as the overlapping swath not contained in the sliding window. In Figure 3.10a, I_4 is the texel image with the greatest overlap with I_6, so the homography H_{46} is found directly between I_4 and I_6. In Figure 3.10b, the homography between I_1 and I_6, $\hat{H}_{16}$, is found by cascading H_{12}, H_{23}, H_{34}, and H_{46}.


H_{12} H_{23} H_{34} H_{45}
I_1 I_2 I_3 I_4 I_5
H_{46}
I_6
H_{12} H_{23} H_{34} H_{45}
I_1 I_2 I_3 I_4 I_5
$\hat{H}_{16}$
H_{46}
I_6

(a) Homographies found directly between texel images.

(b) Cascaded homographies to create a homography estimate between non-adjacent texel images.

Fig. 3.10: Example of cascaded homographies between texel images, including an overlap texel image.

Point projections between two overlapping texel images do not use the pairwise homographies within the sliding window. Overlapping texel images often cover the same area of terrain as other overlapping texel images. Due to this, homography estimates between them can be found directly. This does come with an increased computational complexity, however. The alternative to this is to use the cascaded pairwise homographies and the overlap to sliding window homography estimate for each overlap swath; however, it was found in testing that concatenating homographies can accumulate error, so directly finding the homographies results in a better TDSM registration despite the added computation time.

After the homography estimates and correlations are found, the point projections can be added to the projection matrix. When only considering the texel images inside the sliding window, i.e. the projection matrix used in the non-augmented streaming algorithm, the projection matrix is a $m \times n$ banded block matrix, with m as the sum of points in all texel images, and n texel images. The blocks in this matrix are $[x_{ij}, y_{ij}, \lambda_i j]$ where i is the point and j is the texel image it is projected into.

The projection matrix is banded when only the sliding window texel images are used. The bandedness comes from the look parameter $\mathcal{L}$, where the i^{th} row only finds projections for its points into the texel images $\mathcal{L}$ before and after the owner texel image j; therefore, a row only has non-zero entries in the columns $j - \mathcal{L}$ to $j + \mathcal{L}$.

An example of the sliding window projection matrix, P, with $\mathcal{L} = 1$, is given for six total texel images in the registration. $\mathbf{X}_{ij}$ denotes the points in the i^{th} texel image projected into the j^{th} texel image.

$$P = \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} & 0 & 0 & 0 & 0 \\ \mathbf{X}_{21} & \mathbf{X}_{22} & \mathbf{X}_{23} & 0 & 0 & 0 \\ 0 & \mathbf{X}_{32} & \mathbf{X}_{33} & \mathbf{X}_{34} & 0 & 0 \\ 0 & 0 & \mathbf{X}_{43} & \mathbf{X}_{44} & \mathbf{X}_{45} & 0 \\ 0 & 0 & 0 & \mathbf{X}_{54} & \mathbf{X}_{55} & \mathbf{X}_{56} \\ 0 & 0 & 0 & 0 & \mathbf{X}_{65} & \mathbf{X}_{66} \end{bmatrix} \tag{3.8}$$

The matrix shown in (3.8) only indicates potential non-zero entries, as determined by the look parameter $\mathcal{L}$; if no projection is found between a point in the i^{th} texel image into the j^{th} texel image, the entry in the projection matrix may be zero.

Adding the overlapping texel images matrix makes the projection matrix non-banded. The upper-left corner of the projection matrix is identical to the sliding window projection matrix, and $\hat{\mathcal{M}}$ columns are appended, with $\hat{\mathcal{M}}$ corresponding to how many overlapping images are used for this bundle adjustment optimization, $0 \leq \hat{\mathcal{M}} \leq \mathcal{M}$. There are k rows appended to the bottom of the matrix, with k as the sum of points inside the overlapping texel images. The right-most $\hat{\mathcal{M}}$ columns may have non-zero entries in any row, depending on the point projections, so the augmented projection matrix is non-banded.

An example of the augmented projection matrix, $\hat{P}$, with $\mathcal{L} = 1$ and $\mathcal{M} = 2$, is shown in Equation (3.9).

$$\hat{P} = \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} & 0 & 0 & 0 & 0 & \mathbf{X}_{17} & \mathbf{X}_{18} \\ \mathbf{X}_{21} & \mathbf{X}_{22} & \mathbf{X}_{23} & 0 & 0 & 0 & \mathbf{X}_{27} & \mathbf{X}_{28} \\ 0 & \mathbf{X}_{32} & \mathbf{X}_{33} & \mathbf{X}_{34} & 0 & 0 & \mathbf{X}_{37} & \mathbf{X}_{38} \\ 0 & 0 & \mathbf{X}_{43} & \mathbf{X}_{44} & \mathbf{X}_{45} & 0 & \mathbf{X}_{47} & \mathbf{X}_{48} \\ 0 & 0 & 0 & \mathbf{X}_{54} & \mathbf{X}_{55} & \mathbf{X}_{56} & \mathbf{X}_{57} & \mathbf{X}_{58} \\ 0 & 0 & 0 & 0 & \mathbf{X}_{65} & \mathbf{X}_{66} & \mathbf{X}_{67} & \mathbf{X}_{68} \\ \mathbf{X}_{71} & \mathbf{X}_{72} & \mathbf{X}_{73} & \mathbf{X}_{74} & \mathbf{X}_{75} & \mathbf{X}_{76} & \mathbf{X}_{77} & \mathbf{X}_{78} \\ \mathbf{X}_{81} & \mathbf{X}_{82} & \mathbf{X}_{83} & \mathbf{X}_{84} & \mathbf{X}_{85} & \mathbf{X}_{86} & \mathbf{X}_{87} & \mathbf{X}_{88} \end{bmatrix} \tag{3.9}$$

As with the matrix in (3.8), $\mathbf{X}_{ij}$ denotes a potential non-zero entry in the projection matrix. If there is no projection from the point in the i^{th} texel image to the j^{th} texel image, then the entry is left as zero. The matrix in (3.9) shows an example of the non-banded augmented projection matrix.

Different solving techniques can be used for the augmented projection matrix for running the Levenberg-Marquardt algorithm. In the conventional bundle adjustment, with every texel image and each projection included in a single bundle adjustment using the Levenberg-Marquardt algorithm, the projection matrix is sparse for a very large amount of texel images in the registration, and sparse solving techniques can be used. Streaming bundle adjustment limits the amount of data which can be provided to the matrices used in the Levenberg-Marquardt algorithm, and makes the matrix structure dense. In the augmented projection matrix, similar to the sliding window, the structure is dense and using sparse solving techniques may not result in a reduction in computation time.

3.3 Effect on Processing Time

The additional complexity of the overlap algorithm gives it a higher computational impact when compared with the sliding window approach with no overlap. This additional time complexity comes in four main areas:

1. Calculating convex hulls for each texel image in flight data.

2. Creating an overlap instruction set for streaming bundle adjustment.
3. Loading in overlap frame data from memory, finding the homography estimates for the overlap swaths and computing the projections into neighboring texel images.
4. Computations for Levenberg-Marquardt bundle adjustment with the larger projection matrix.

Calculating the convex hulls for the entire data set takes significantly longer, but is only required to be calculated once for each data set. The bottleneck on calculating the convex hulls is in read/write speed on disk to extract and load the texel image data into memory. Texel image data for each frame is saved in a compressed file format on disk, and calculating the convex hull requires extracting and loading in this data so that the X, Y, Z points are able to be retrieved from this data. This is dependent on hard drive speed. The time complexity for this step is $\mathcal{O}(n)$, for n texel images, and is only required to be done once for each data set. After the convex hulls are calculated, the points are saved in a .dat file which is much quicker to load in than the full data for each texel image. Pre-calculating and saving the convex hulls saves the time of loading in each texel image every time it is used.

Using the convex hulls to create the overlap instruction set is another time complexity addition. For each frame in the sliding window, the intersection with every other frame must be calculated in order to find if any frames intersect, and if so, sort frames in descending order of greatest intersection with the main sliding window. The convex hulls are saved in a binary tree so that the intersection of every texel image does not need to be calculated, so finding the overlapping frames for a single sliding window optimization has a time complexity of $\mathcal{O}(m \log n)$, for n total frames in the flight data and m sliding window frames.

Another addition which slows down the post-processing computations is in using the overlap instruction sets. For each bundle adjustment, the overlapping frames must be loaded in from the texel image data saved on disk. This only requires loading a maximum of $\mathcal{M}$

frames, where $\mathcal{M}$ is defined as the maximum frame parameter. The additional time taken due to this step is small, relative to the additions in calculating and using the convex hulls.

The time complexity of the Levenberg-Marquardt algorithm compared to streaming bundle adjustment cannot easily be determined analytically, as it is dependent on the matrix structure and solving methods used. It is instead experimentally analyzed in Section 4.3.

To illustrate the difference between the streaming bundle adjustment algorithm, developed by Khatiwada [1], and the modified streaming bundle adjustment algorithm which uses overlap, block diagrams are shown below. The streaming bundle adjustment is shown in Figure 3.11. The final output in Figure 3.11, Reconstructed TDSM With Optimized 3D Points, is the reconstructed terrain after the streaming bundle adjustment has corrected for sensor error. The flow diagram for the augmented streaming bundle adjustment algorithm is shown in Figure 3.12.

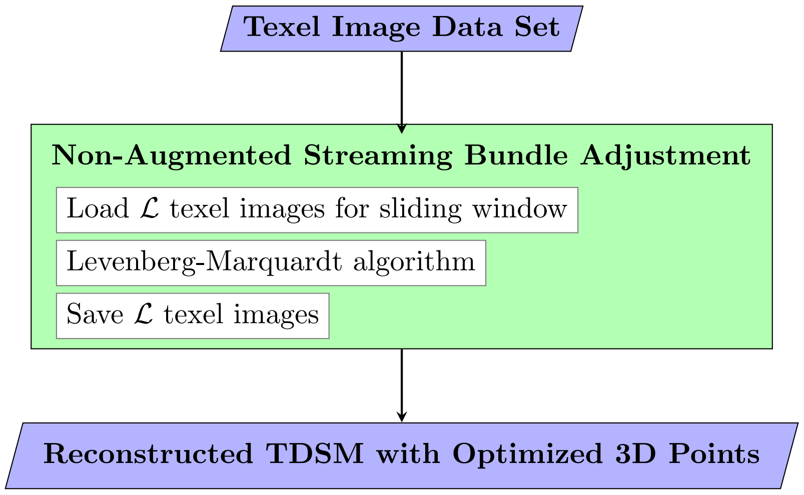

Texel Image Data Set
Non-Augmented Streaming Bundle Adjustment
Load $\mathcal{L}$ texel images for sliding window
Levenberg-Marquardt algorithm
Save $\mathcal{L}$ texel images
Reconstructed TDSM with Optimized 3D Points

Fig. 3.11: Block diagram for the non-augmented streaming bundle adjustment algorithm.

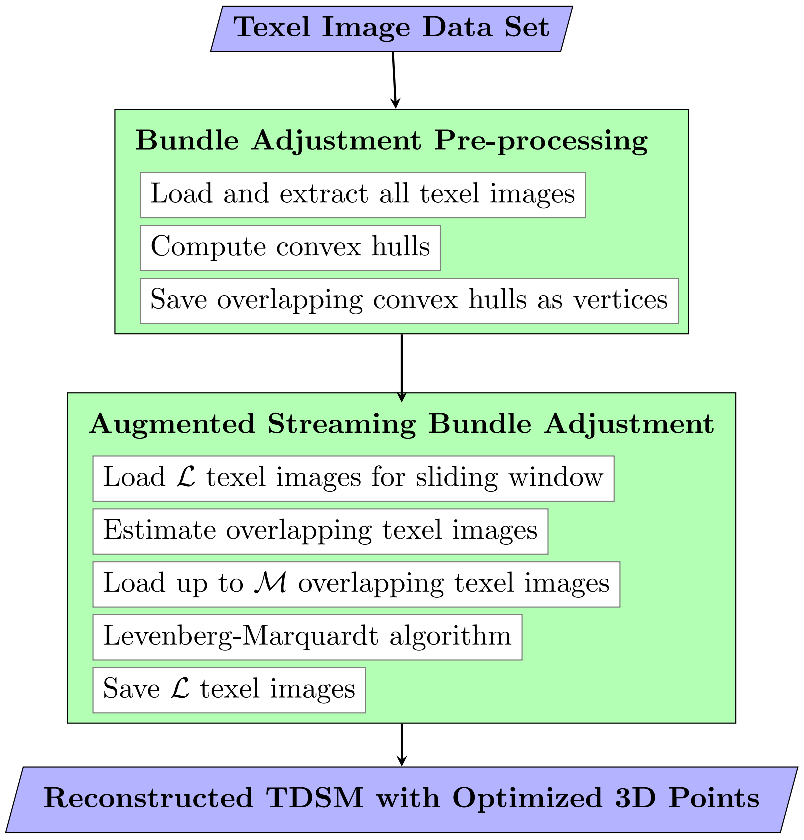

Texel Image Data Set
Bundle Adjustment Pre-processing
Load and extract all texel images
Compute convex hulls
Save overlapping convex hulls as vertices
Augmented Streaming Bundle Adjustment
Load $\mathcal{L}$ texel images for sliding window
Estimate overlapping texel images
Load up to $\mathcal{M}$ overlapping texel images
Levenberg-Marquardt algorithm
Save $\mathcal{L}$ texel images
Reconstructed TDSM with Optimized 3D Points

Fig. 3.12: Block diagram for the augmented streaming bundle adjustment, as implemented in this research.

CHAPTER 4

Results

The methods presented in this work were tested on a set of texel images to test the feasibility of running this algorithm and analyze any improvement in the resulting TDSMs. The data set used in this research was acquired from a sUAV flight over Utah State University campus on 28 July 2023. The portion of the flight path used in testing this data is shown in Figure 4.1.

The sUAV captured texel images from a height of 60 meters, at a rate of five texel images per second. The full flight contains 2394 texel images.

4.1 A Metric for Quantifying Accuracy

To test the augmented streaming bundle adjustment and verify the effectiveness of it compared to conventional bundle adjustment and streaming bundle adjustment, a method for comparing the effectiveness and accuracy of different registered TDSMs must be considered.

The results should be measured by how close the resulting TDSM is to the actual terrain being reconstructed. A highly accurate and reliable method for determining the accuracy of the TDSM would be comparing it to the ground truth [25, 26]. However, for the data acquired from the sUAV and used in this research, the ground truth data was not acquired and thus is not available, so different TDSM registrations cannot be compared to an objective ground truth to determine accuracy. Other qualitative and quantitative methods for determining the effectiveness of the final TDSM registration must be used.

One such quantitative method is by comparing the sum of squared error after each optimization of the Levenberg-Marquardt algorithm. The program used in this research can store the factor by which the sum of squared error is improved from the initial error to after the Levenberg-Marquardt algorithm. In the results presented here, the factor is

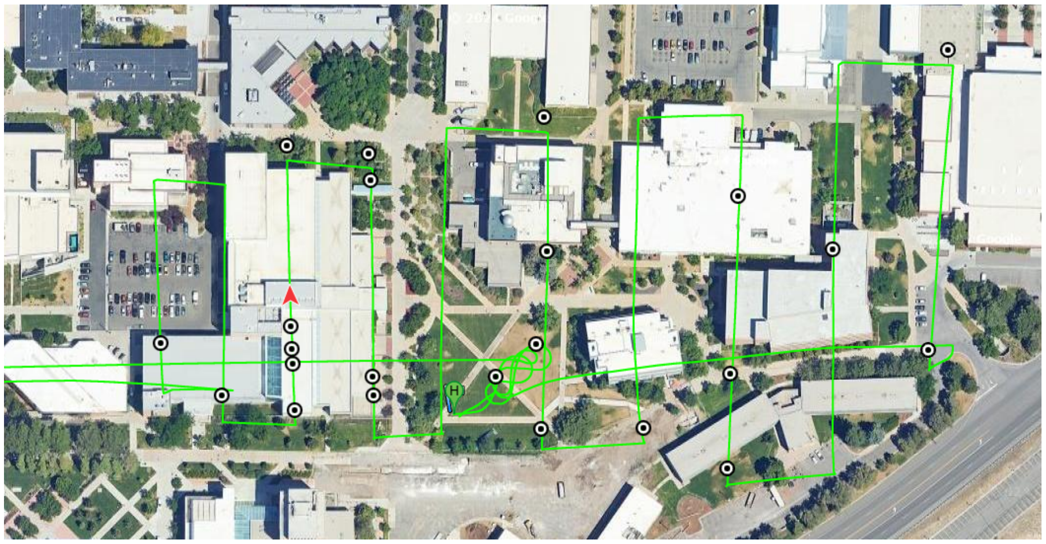


Fig. 4.1: Aerial image of the sUAV flight pattern used. Created with DJI Flight Log Viewer.

normalized by the number of points used in the optimization, so that the non-augmented and augmented cases can be directly compared. A higher number for this factor indicates that the Levenberg-Marquardt algorithm was able to correct for more error in the initial poses, and converge to a result which better matches the model based on the point projections.

The sum of squared error is based on the point projections found and stored in the projection matrix; so if the projections are accurate to the true model of the captured data, then a larger factor of improvement in the error makes the resulting TDSM more accurate. However, this method may be prone to errors. If the point projections found are incorrect, then the model that it is fitting is worse than the true model and a higher factor of reducing the sum of squared error does not indicate an increase in accuracy.

The point projections being incorrect can usually happen from the homography estimates not accurately matching the true model, resulting in poor correlations. The chance of this happening is reduced by setting a high confidence parameter for RANSAC, when finding the homography between two texel images. Additionally, this is checked by setting the threshold for a point correlation to be found. If the normalized cross correlation cannot find a point, then the projection is not saved and the low confidence point does not affect the model.

Another qualitative method of determining accuracy in the resulting TDSM is by visually inspecting the resulting TDSM. The 3D point cloud and optical image can be analyzed for any discrepancies or trends that would indicate that the model is not accurate. This can be done based on the assumptions of the terrain being captured; e.g. for a flat surface, an accurate TDSM would have multiple texel images which cover this surface to be level with respect to each other and have minimal variation in angles and range, or for a captured building, the overlaid image should closely align to the corner of the building. In this research, both the point cloud and the TDSM with an overlaid image are analyzed and compared to verify the quantitative results of the sum of squared error.

4.2 Effect of Augmented Bundle Adjustment Parameters

The two parameters tested in the augmented bundle adjustment are ρ, the minimum overlap percent parameter, and $\mathcal{M}$, the maximum number of texel images to augment the sliding window with. In testing, the control case for the streaming bundle adjustment was using $\mathcal{M} = 0$, so that only the sliding window texel images are used for the non-augmented images.

The first case for which the augmented bundle adjustment is tested is in a set of consecutive texel images by analyzing the performance for a consecutive set. This is testing the assumption that the extra data from overlapping texel images outside of the sliding window is able to improve the sliding window registration. The non-registered set of texel images used in this testing is shown in Figure 4.2, with the point cloud viewed from the side, shown in Figure 4.3. This shows the uncorrected error from the camera poses at the time the texel images were captured, which causes the discrepancy between the texel images. The goal of TDSM registration is to correct for the errors and remove the aberrations present so that after registration, the errors will be corrected and better match the true terrain representation.

The 3D points in point clouds shown are colored according to the elevation relative to the highest and lowest points in a texel image. For example, in Figure 4.3, the blue points shown on the left side of the image correspond to a higher elevation on the roof of a building, and the yellow and orange points seen at the bottom of the image correspond to a lower elevation on the ground.

This was tested on a set of 60 texel images with $\rho = 0.15$, $\mathcal{M} = 5$, $\mathcal{L} = 5$. The sum of squared error distance improvement is presented in Table 4.1. Each iteration had $\hat{\mathcal{M}} = 5$, with $\hat{\mathcal{M}}$ defined as the amount of texel images which meet the rho parameter and are added to the bundle adjustment, $0 \leq \hat{\mathcal{M}} \leq \mathcal{M}$.

Optimization	**Non-augmented**	**Augmented**	**Augmentation Improvement**
1	0.0050	0.0275	5.50
2	0.0113	0.0232	2.035
3	0.0102	0.0065	0.637
4	0.0075	0.0061	0.813
5	0.0074	0.0078	1.054
6	0.0065	0.0079	1.215
7	0.0189	0.0260	1.376

Table 4.1: Augmentation improvement for each optimization in bundle adjustment. The non-augmented and augmented numbers represent the sum of squared error, normalized to the number of points present in the registration.

The augmentation improvement represents the factor by which the sum of squared error improves, when comparing the non-augmented and augmented bundle adjustment. A single optimization, as shown in Table 4.1, is the Levenberg-Marquardt algorithm running on a single sliding window, so for a single optimization, $3\mathcal{L} + \hat{\mathcal{M}}$ texel images are used in processing, and $\mathcal{L}$ texel images are saved as the maximum likelihood estimate after each optimization.

The results presented are normalized to the number of points inside each bundle adjustment. Table 4.1 shows, when testing the augmented bundle adjustment, the sum of squared error is improved for every iteration except two, and thus, a greater amount of errors are corrected to better fit the model based on the point projections. To verify that the resulting TDSM is not fitting to an incorrect model, the TDSMs are compared in Figure 4.4 and Figure 4.5. Visually, the two resulting TDSMs with overlaid images are similar, and both represent an improvement over the individual texel images with no error correction,

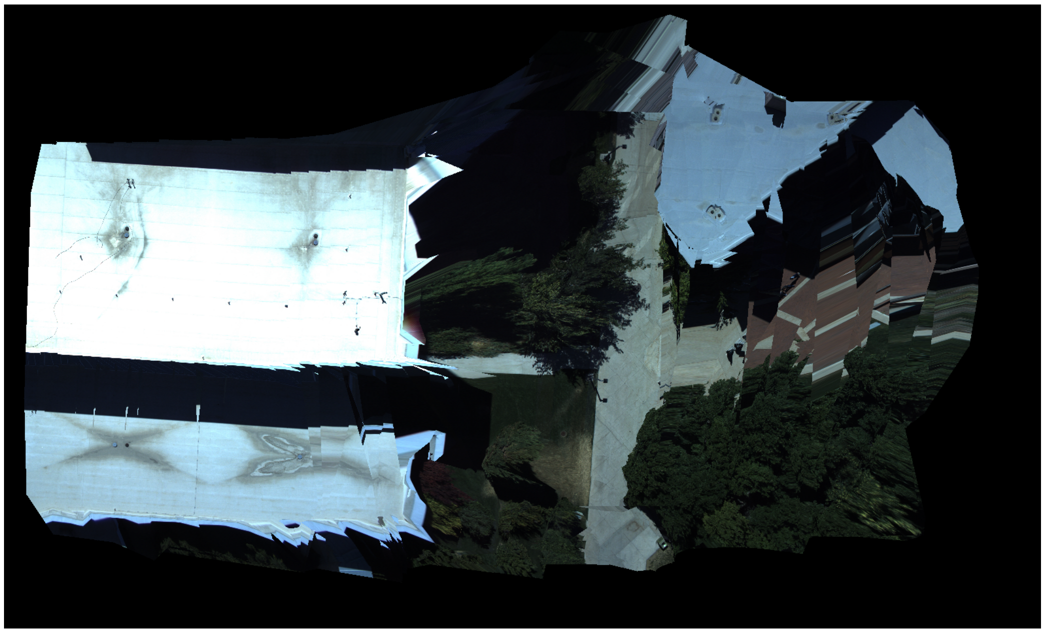


Fig. 4.2: Set of texel images before processing. The discrepancies in the courtyard on the right shows how the the uncorrected camera poses contain error.

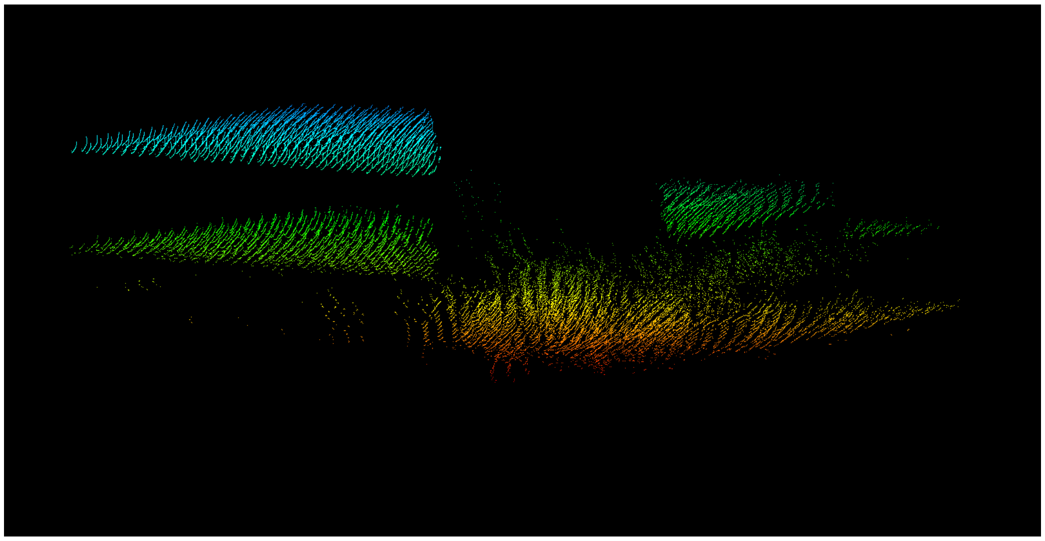


Fig. 4.3: Point cloud of all texel images before processing. Axis aligned view. Variations in the camera poses cause the different texel images to not be aligned to the same plane.

from Figure 4.2.

The differences are more prominent when viewing the registered TDSM point clouds aligned with the axis, Figure 4.6 and Figure 4.7. The improvements in both point clouds can be seen clearly when compared with the original point clouds containing pose error, in Figure 4.3. The surfaces are more closely aligned to a single plane, which can be seen on the left side with the top of the two buildings. The differences between the non-augmented and augmented TDSMs can also be seen in these point clouds. The non-augmented bundle adjustment has a curve upwards, most noticeable on the surface of roofs, on the left side of the image. In contrast, the roof in the augmented bundle adjustment is more flat; this is because the overlapping texel images used in the registration help to correct for this error and align the surfaces to a single plane.

Visually inspecting the above figures aligns with the results from the sum of squared error distances in each bundle adjustment optimization, as the TDSM resulting from the augmented algorithm shows an improvement when compared with the non-augmented algorithm.

Results from another TDSM registration are presented in Table 4.2. This TDSM corresponds to another set of 60 texel images in the data set. Each optimization had $\hat{\mathcal{M}} = 5$ overlapping texel images to augment the matrix, and the same parameters as above, with $\rho = 0.15$, $\mathcal{M} = 5$, $\mathcal{L} = 5$.

Optimization	**Non-augmented**	**Augmented**	**Augmentation Improvement**
1	0.00025	0.000422	1.553
2	0.00056	0.00066	1.168
3	0.0010	0.00064	0.604
4	0.0028	0.0036	1.275
5	0.0036	0.0682	18.82
6	0.16	0.3207	1.99
7	1.20	0.83	0.684

Table 4.2: Augmentation improvement for each optimization in bundle adjustment.

As Table 4.2 shows, by using the sum of squared error metric, the performance of

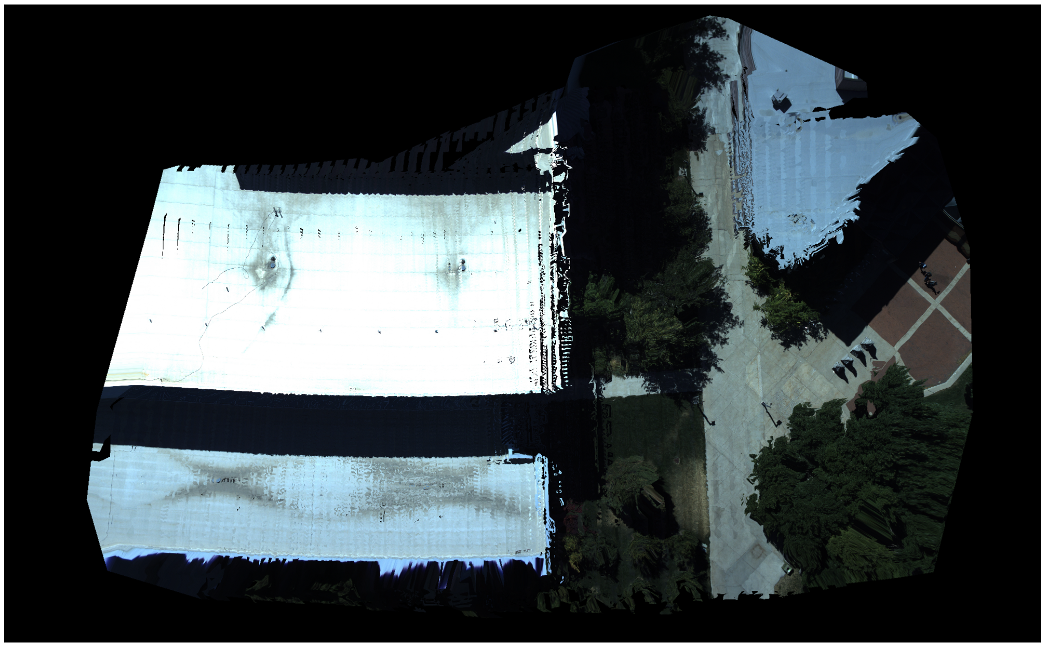


Fig. 4.4: TDSM registration using non-augmented bundle adjustment.

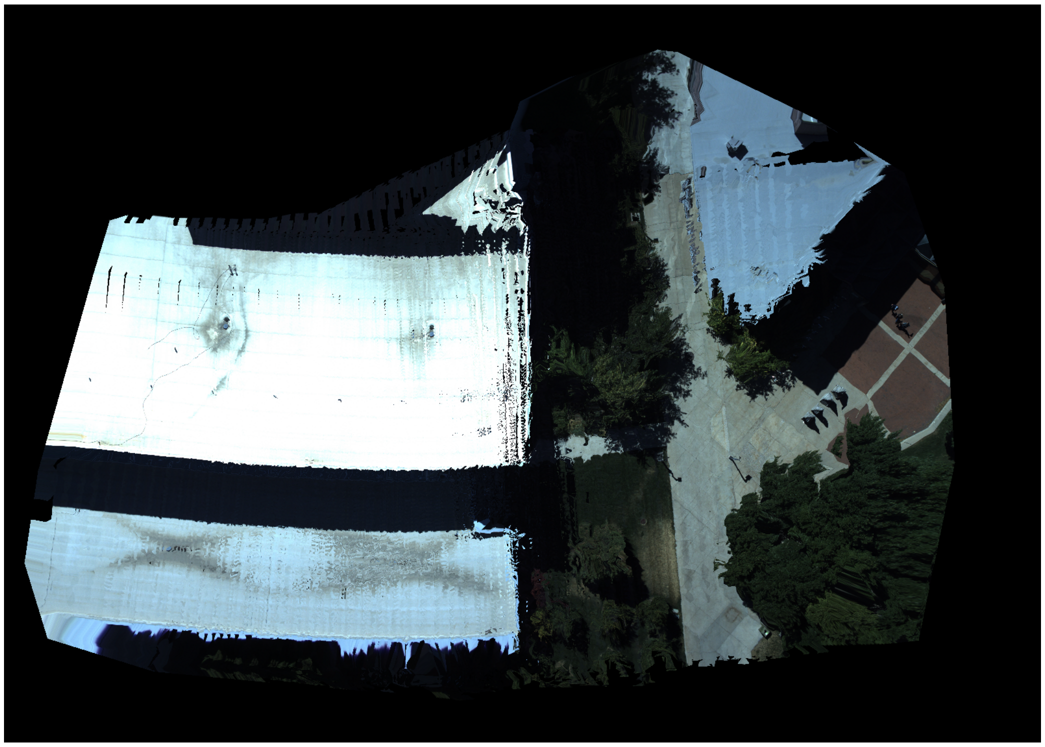


Fig. 4.5: TDSM registration using augmented bundle adjustment.

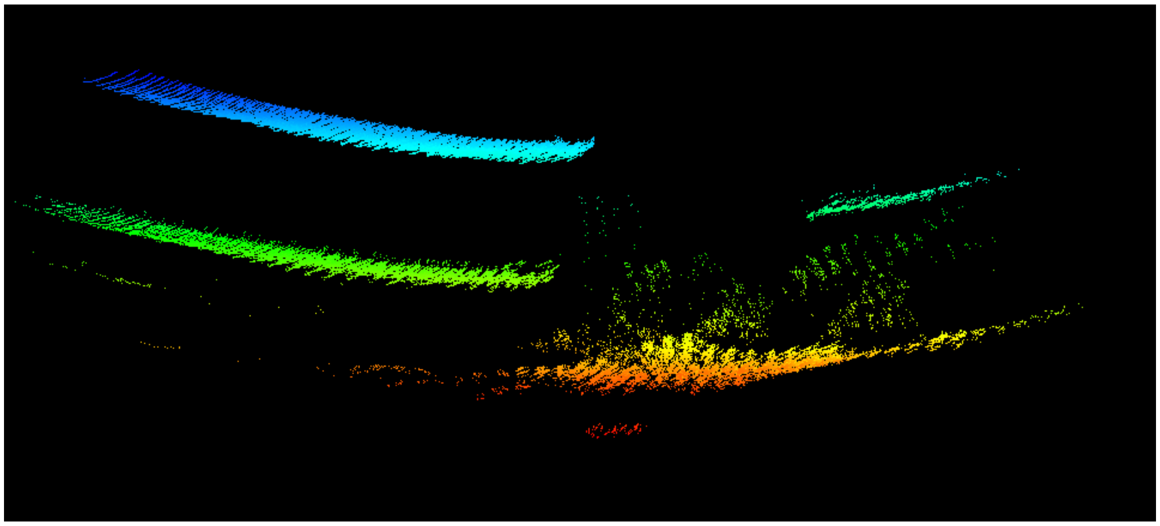


Fig. 4.6: Point cloud after TDSM registration using non-augmented bundle adjustment. Errors in the initial camera poses cause curvature in the registered TDSM.

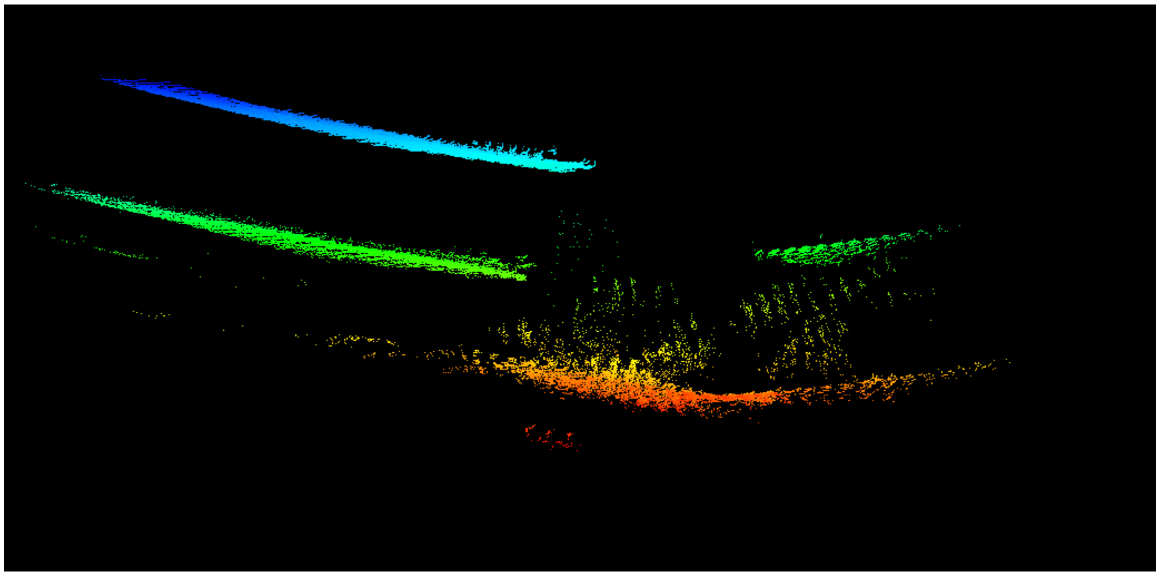


Fig. 4.7: Point cloud after TDSM registration using augmented bundle adjustment. The points and camera poses are corrected to be more closely aligned to a single plane, which better matches the true terrain representation.

the augmented bundle adjustment is better in most iterations, significantly better in some iterations, and never significantly worse. A comparison for both registered TDSMs is shown in Figure 4.8 and Figure 4.9, for the non-augmented and augmented TDSM registrations, respectively.

Both represent an improvement over the texel images without any error correction applied. However, there are some differences visible between the TDSMs, such as the sidewalk on the right hand side of the figure, and the top of the building on the left side of the figure. In these places, the augmented bundle adjustment better corrects for error and aligns closer to the expected terrain. The additional data used helps the Levenberg-Marquardt algorithm find a better maximum likelihood estimate of the poses. In addition, the point cloud is noticeably different between each, shown in Figure 4.10.

The axes are aligned to the ENU coordinate system in Figure 4.10, with north-south on the vertical axis, and east-west on the horizontal axis. For this terrain, from referencing the known flight pattern from the sUAV previously shown in Figure 4.1, the true representation of the terrain is with the building aligned north-south.

Using non-augmented bundle adjustment, the reconstructed TDSM, shown in Figure 4.10(a), is misaligned with the true representation. The most noticeable example of this can be seen in the corners of the building toward the center of the figure, which do not align to a single edge and cause sharp edges after triangulation is done on the TDSM. Another example can be seen in the sidewalk on the upper right side of the figure, which is not aligned to a straight line. This happens due to the limited sliding window used in choosing the texel images in bundle adjustment. However, when overlap frames are used in the augmented case, shown in Figure 4.10(b), the additional data helps to align the data to the true representation in the world coordinate system.

In both of the TDSM registrations presented in this thesis, different ρ values were tested. For a lower ρ, no change was found. At $\rho = 0.15$ a sufficient number of texel images were found to meet the $\mathcal{M}$ parameter, and the overlapping texel images are sorted according to highest overlap, so a lower value chosen for ρ does not include any different texel images

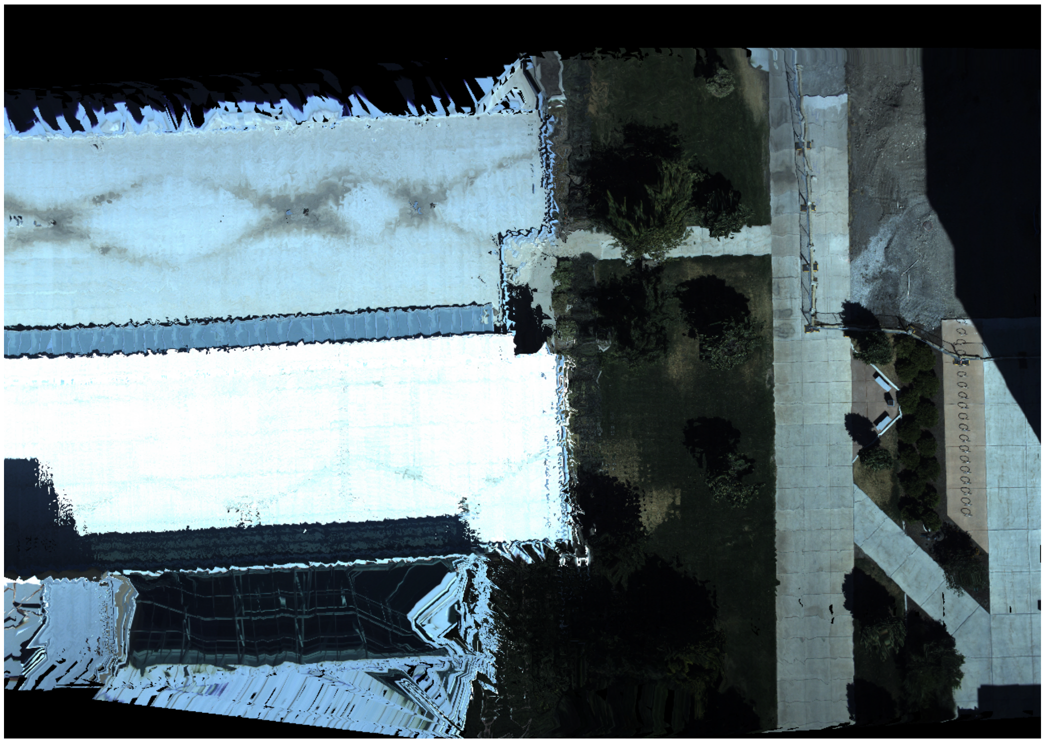


Fig. 4.8: TDSM registration using non-augmented bundle adjustment. Some discrepancies are noticeable in the edges of the building in the center, and on the sidewalk to the right.

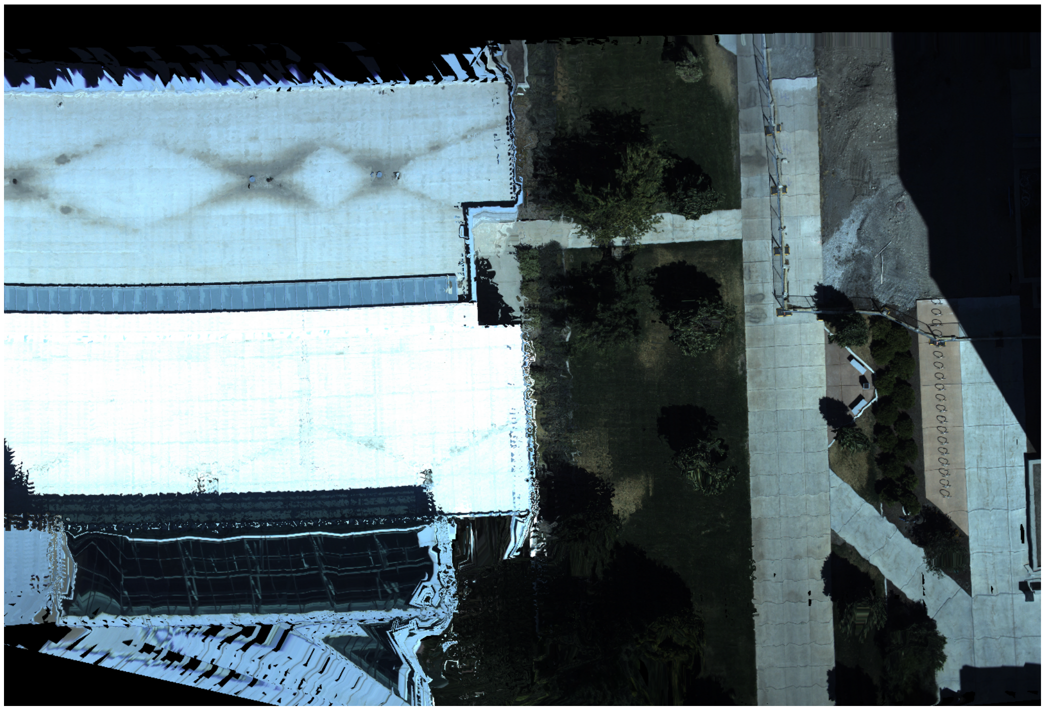
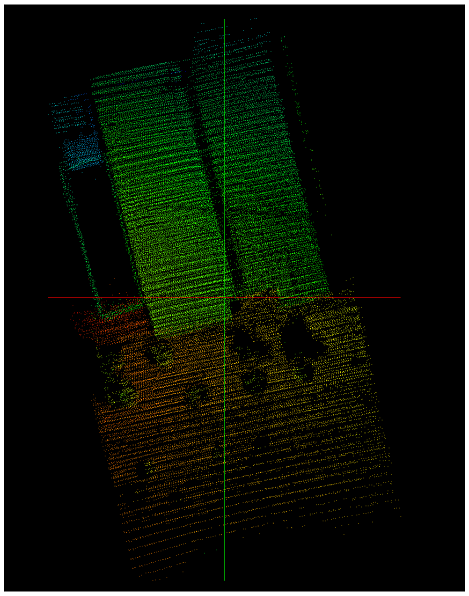


Fig. 4.9: TDSM registration using augmented bundle adjustment. The discrepancies seen in the non-augmented TDSM registration are better corrected for.


(a) Non-augmented.

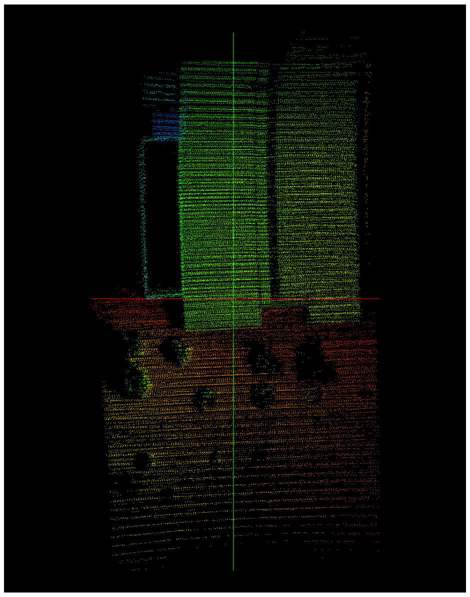

(b) Augmented.

Fig. 4.10: Registered point clouds using non-augmented and augmented bundle adjustment. The point clouds are aligned in the ENU coordinate system. When compared with the non-augmented bundle adjustment, the augmented registration is closely aligned north-south, which better matches the true flight path.

in the augmented bundle adjustment. It was found, for both TDSM registrations presented, increasing ρ had no effect until $\rho >= 0.40$, in which case no overlapping texel images met the minimum ρ and performance was identical to the non-augmented streaming bundle adjustment.

Because $\mathcal{M}$ defines the limit of how many additional texel images are potentially added to the bundle adjustment, it also defines the computational and memory increase. The computational differences are analyzed in Section 4.3. An unrestricted $\mathcal{M}$, meaning that there is no limit to the number of overlapping swaths added to the bundle adjustment, is not recommended. Unless the flight pattern is verified to not have any areas that have a very large number of overlapping texel images, it may add far more overlapping texel images than sliding window texel images, making the program infeasible to run.

Similarly, a very low ρ parameter is not recommended. The convex hull checked is the union of all texel images within the sliding window, which may consist of a large number of texel images and thus, a wide area of terrain to check for overlap. Setting the ρ parameter to very low will cause texel images with minimal overlap with the sliding window of terrain to be added in the overlap algorithm. If there is minimal overlap there is a high likelihood that an accurate homography will not be found before creating the projection matrix, so in any case the texel images will get thrown out and not affect the final registration. This still requires loading the texel images, extracting the data from disk, and performing feature detection on the EO images, so if the texel images are discarded before bundle adjustment, the algorithm runs significantly slower for no change or benefit in the final registration. For this reason, setting ρ to be equal to or greater than 0.10 is recommended.

TDSM registrations from sections of the entire flight were tested. When there were at least $\mathcal{M} = \mathcal{L}$ overlapping texel images in each bundle adjustment, the results are consistent with the results presented in this section in that a greater improvement in error was noticed. Due to the flight pattern, there were some sections in the data for which there were no overlapping swaths; in such cases, performance was identical to the streaming bundle adjustment. For implementation and use of this algorithm, a consideration for future data acquisition flights is to determine a flight pattern such that each texel image has at least one non-adjacent texel image which meets the minimum ρ.

In testing, it was found that $\rho = 0.10$ is the minimum value to find a useful amount of point projections. With a higher overlap, there are more point projections visible in neighboring texel images and thus, more data points which help the bundle adjustment correct for errors. For capturing future data sets, a flight pattern that has a minimum overlap of $\rho = 0.20$ is recommended to increase the accuracy of this algorithm.

4.3 Computation Time

As the primary motivation and focus of this research is in utilizing low-cost hardware in TDSM registration, the feasibility of running this algorithm on commonly available hardware is considered. The two metrics to measure the difference in the algorithm are the total runtime and memory usage, which are both compared to the streaming bundle adjustment to measure its effectiveness. This section contains experimental results. Section 3.3 has an explanation of the theoretical computational increases.

The first step is to pre-compute the convex hull for each texel image in the data set. Loading the entire set of texel images to compute the convex hulls is not required in the non-augmented streaming bundle adjustment, so any increase in this step is strictly additional time for the overlap algorithm. The computation from this comes mainly from loading the compressed data from memory and extracting it; this results in the bottleneck being the hard drive read and write speed. The hardware used to test this algorithm is a Samsung 850 EVO 500GB SSD. In testing, the SSD had an average read/write speed of 80MB/s and 130MB/s, respectively. This step was tested with different sizes of texel images to test the performance on large sets of data. The results for memory usage and runtime are shown in Table (4.3).

Texel Images	**Memory** *(Mb)*	**Runtime** *(seconds)*	**Time per image** *(seconds)*
240	309	71.8	0.29
480	390	196.8	0.41
1200	446	624.1	0.52
2400	414	1168.1	0.49

Table 4.3: Runtime and memory usage for extracting texel images and computing convex hulls.

The jump in time per image between 240 and 480 texel images is due to hard drive read/write speed. At 480 texel images, the hard drive bottleneck is reached and the hard drive consistently has 100% read/write usage when the algorithm is running. Before this bottleneck, CPU multithreading is the limiting factor, so there is a lower time per image when using 240 texel images. From 480 texel images onward, memory usage and time per texel image remains constant when more texel images are added to the algorithm.

The additional time in this step is a strict increase over the non-augmented streaming algorithm. However, generating the convex hulls is only required to be done once for each data set. If this remains a bottleneck in future work, a potential area of increase would be saving the convex hulls when generating the texel images from the raw data. This would skip the step of loading and extracting each texel image from the compressed data stored on the hard drive.

The other computational increases come from loading overlapping texel images and performing the Levenberg-Marquardt algorithm with larger matrices. To test this, the TDSM registration algorithm was run on the same set of texel images using the streaming bundle adjustment and the augmented streaming bundle adjustment with overlapping texel images. The algorithm was tested on a desktop computer running Windows 10 Enterprises with an AMD Ryzen 7 1700 processor, an NVIDIA GeForce GTX 1050, and 24 GB of RAM. Optimizations to use the GPU via NVIDIA's CUDA toolkit were used within the algorithm.

The TDSM registration was tested on the same set of 60 texel images, with $\mathcal{L} = 7$, and the $\mathcal{M}$ parameter is adjusted to view the impact that the overlap algorithm has on the total runtime. The results for each are compared in Table 4.4.

$\mathcal{M}$	**Memory** *(Gb)*	**Total Runtime** *(seconds)*
0	4.1	396.9
3	5.2	664.9
5	5.6	901.0
7	6.1	1280.3

Table 4.4: Runtimes and memory usage for TDSM registration with varying $\mathcal{M}$.

The $\mathcal{M} = 0$ row in Table 4.4 is identical to the non-augmented algorithm, which the performance of the augmented algorithm can be compared to.

A very high $\mathcal{M}$ may be infeasible depending on the flight pattern, if there are a large amount of overlapping texel images which the hardware is unable to store in memory; the parameter was $\mathcal{M}$ was tested less than or equal to $\mathcal{L}$ for optimal results in the final registration.

The results presented in Table 4.4 show that the maximum overlap images parameter $\mathcal{M}$ has a large impact on the runtime. This occurs because adding overlap images requires each sliding window to check memory to see if the texel image currently has data in memory, and if not, load and extract the data for the corresponding texel image from where it is saved on disk. For each overlap image, the homographies need to be found, both to the sliding window image which has shares the highest overlap with it, and also checking if a homography exists for each other overlap image in the optimization. This results in a large slowdown when a higher $\mathcal{M}$ is used. A potential optimization that could reduce this bottleneck, which was not implemented in this research, is to keep more overlapping texel images in memory than are strictly necessary. For example, keeping the last several texel images used in memory, even if they are not required in the current iteration. Although this would increase the memory requirements, it eliminates the time needed to load and extract the data for the overlapping texel images from the hard drive.

Adding more overlap swaths also results in a slowdown for the Levenberg-Marquardt algorithm. Although the matrix is still dense, and dense solving techniques can be used for the matrix equations, the computations for solving the matrices take longer. This can be adjusted by using another parameter developed as part of the original bundle adjustment algorithm, which is the maximum number of iterations inside the Levenberg-Marquardt algorithm. Setting the number of Levenberg-Marquardt iterations to be lower reduces the likelihood of finding a lower total error, but also reduces the computations required.

CHAPTER 5

Conclusion

The streaming bundle adjustment makes creating a single TDSM from large amounts of texel images feasible by limiting the maximum amount of data which is processed at once. The developments presented in this document build upon that research to correct for one of the weaknesses in streaming bundle adjustment, which is that only time-adjacent texel images are used in bundle adjustment.

The research presented in this document shows that texel images from the entire data set, instead of only neighboring texel images, can be used to improve the registered TDSMs. The development of efficient overlap estimation methods and an augmented bundle adjustment algorithm using the estimated terrain overlap better corrects for accumulated errors between texel images captured from different times in the flight. Testing on real-world data acquired from a sUAV shows that the resulting TDSMs are more accurate and contain less error when using the augmented bundle adjustment, when compared to the non-augmented bundle adjustment.

There are many avenues which future research would be able to build upon the advancements presented here. The resulting TDSMs show improvements in addressing the error accumulation over time in the loop-closing problem, so that sections of the flight which are far apart in time are accurately registered to each other in a single TDSM. This can be applied to a much larger set of texel images in a single registration. This research focused on a relatively small section of texel images; the typical test set used in this document was 60 texel images, which represents around 12 seconds of acquired data from a sUAV flight. The full flight, which was eight minutes in the data set used here, can use the optimizations presented here to register into a single TDSM that is much closer to the true terrain representation.

The research here still assumes that the texel images within the sliding window are consecutive in time. One direction for improving upon this research would be to remove any assumption about the order of given texel images, and optimally partition them into registration iterations for the highest overlap which can then be corrected in bundle adjustment. This would remove the assumption that the data is arranged according to neighboring texel images and allow for greater flexibility in data acquisition, e.g. using texel images acquired from many sUAV's capturing the same area of terrain.

Another method would be adapting this algorithm to run in real-time. In this research the assumption was made that the entire data set from the flight is available at the time of post-processing. Improving on this, such as streaming data from the sUAV to a ground station for registering a TDSM while data is being acquired, would require adapting the algorithm to compute convex hulls and calculate the optimal overlap while texel images are being constantly added to the data set. This could be used in applications where it is important to have an accurate TDSM immediately after the flight, instead of waiting for post-processing to complete.

The augmented bundle adjustment algorithm maintains the advantages of the streaming bundle adjustment by being optimized and efficient, capable of processing large amounts of data on low-cost hardware. The augmented bundle adjustment is still fast and uses a limited amount of memory, and results in TDSMs which have better error correction.

REFERENCES

[1] B. Khatiwada, “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),” Ph.D. dissertation, Utah State University, Logan, UT, 2023.

[2] C. Kapoutsis, C. Vavoulidis, and I. Pitas, “Morphological iterative closest point algorithm,” *IEEE Trans. Image Process.*, vol. 8, no. 11, pp. 1644–1646, 1999.

[3] R. Kataoka, R. Suzuki, Y. Ji, H. Fujii, H. Kono, and K. Umeda, “ICP-based SLAM using lidar intensity and near-infrared data,” in *2021 IEEE/SICE International Symposium on System Integration (SII)*, 2021, pp. 100–104.

[4] A. Myronenko and X. Song, “Point set registration: Coherent point drift,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, 2010.

[5] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

[6] J.-Y. Han, C.-S. Chen, and C.-T. Lo, “Time-variant registration of point clouds acquired by a mobile mapping system,” *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 1, pp. 196–199, 2014.

[7] B. Zhu, L. Zhou, S. Pu, J. Fan, and Y. Ye, “Advances and challenges in multimodal remote sensing image registration,” *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 4, no. 2, pp. 165–174, 2023.

[8] T.-A. Teo and S.-H. Huang, “Automatic co-registration of optical satellite images and airborne lidar data using relative and absolute orientations,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 5, pp. 2229–2237, 2013.

[9] J. Lee, X. Cai, C.-B. Schönlieb, and D. A. Coomes, “Nonparametric image registration of airborne lidar, hyperspectral and photographic imagery of wooded landscapes,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 11, pp. 6073–6084, 2015.

[10] R. M. Palenichka and M. B. Zaremba, “Automatic extraction of control points for the registration of optical satellite and lidar images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 7, pp. 2864–2879, 2010.

[11] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[12] Z. Zheng, Y. Li, and W. Jun, “Lidar point cloud registration based on improved ICP method and SIFT feature,” in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2015, pp. 588–592.

[13] X. Ge, H. Hu, and B. Wu, “Image-guided registration of unordered terrestrial laser scanning point clouds for urban scenes,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 9264–9276, 2019.

[14] T. C. Bybee, “An automatic algorithm for textured digital elevation model formation using aerial texel swaths,” Master of Science, Utah State University, Logan, UT, 2015.

[15] T. C. Bybee and S. E. Budge, “Method for 3-D scene reconstruction using fused lidar and imagery from a texel camera,” *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8879–8889, 2019.

[16] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, “Bundle adjustment in the large,” in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II 11*. Springer, 2010, pp. 29–42.

[17] B. Chamberlain and S. E. Budge, “LiDAR scene reconstruction using imagery and frame overlap from a Texel camera,” in *Laser Radar Technology and Applications XXIX*, G. W. Kamerman, L. A. Magruder, and M. D. Turner, Eds., vol. 13049, International Society for Optics and Photonics. SPIE, 2024, p. 130490C. [Online]. Available: https://doi.org/10.1117/12.3013914

[18] B. M. Boldt, S. E. Budge, R. T. Pack, and P. D. Israelsen, “A handheld texel camera for acquiring near-instantaneous 3D images,” in *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, 2007, pp. 953–957.

[19] M. Aboutalebi, A. F. Torres-Rua, M. McKee, W. Kustas, H. Nieto, and C. Coopmans, “Validation of digital surface models (DSMs) retrieved from unmanned aerial vehicle (UAV) point clouds using geometrical information from shadows,” in *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping IV*, J. A. Thomasson, M. McKee, and R. J. Moorhead, Eds., vol. 11008, International Society for Optics and Photonics. SPIE, 2019, p. 110080L. [Online]. Available: https://doi.org/10.1117/12.2519694

[20] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to sift or surf,” in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[22] M. Aboutalebi, A. F. Torres-Rua, M. McKee, W. Kustas, H. Nieto, and C. Coopmans, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 2019.

[23] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, 2000, pp. 602–614.

[24] S. Gillies *et al.*, "Shapely: manipulation and analysis of geometric objects," toblerity.org, 2007–. [Online]. Available: https://github.com/Toblerity/Shapely

[25] A. Godil, R. Bostelman, K. Saidi, W. Shackleford, G. Cheok, M. Shneier, and T. Hong, "3D ground-truth systems for object/human recognition and tracking," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 719–726.

[26] C. M. de la Osa, G. G. Anagnostopoulos, M. Togneri, M. Deriaz, and D. Konstantas, "Positioning evaluation and ground truth definition for real life use cases," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016, pp. 1–7.