`filters.als_tpu`: A PDAL Filter for Airborne Laser Scanning Total Propagated Uncertainty

Preston Hartzell
University of Houston
pjhartzell@uh.edu

December 1, 2021

Abstract

`filters.als_tpu` generates per-point total propagated uncertainty for airborne laser scanning point clouds without requiring access to sensor measurement data or knowledge of the sensor model geometric conventions. The fundamental product of the filter is a 3×3 covariance matrix for each point. The filter implements a standard form of the lidar georeferencing equation based on a generic set of measurements and geometric conventions. This technical document details the georeferencing model, measurements, and geometric conventions employed in the georeferencing model. Variance propagation is also reviewed and the general steps employed in the code are outlined.

1 Background

Three components are required to generate per-point total propagated uncertainty (TPU) for airborne laser scanning (ALS) point cloud data when applying traditional variance propagation:

1. The ALS measurements, or observations, used in the airborne lidar georeferencing equation to compute the ground coordinates of each point.
2. The uncertainty, i.e., variance or standard deviation, associated with each ALS measurement.
3. The ALS system geometric conventions.

The first component consists of lidar range, scanner angle, inertial motion unit (IMU) position (latitude, longitude, height or x, y, z), IMU attitude (roll, pitch, heading), axes misalignment between the scanner and IMU frames (boresight roll, pitch, yaw), and location of the laser emission point in the IMU frame (lever arm x, y, z). The second component is simply the uncertainty for each of these measurements. The third component, the ALS system geometric conventions, enables us to form an airborne lidar georeferencing equation that is mathematically consistent with the ALS measurements.

To better explain what is meant by "geometric conventions", we start by defining a standard form of the airborne lidar georeferencing equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{ground}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{trajectory}} + \boldsymbol{R}_{\text{imu}} \left(\boldsymbol{R}_{\text{bore}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{scanner}} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{lever}} \right), \tag{1}$$

where the "ground" vector is the georeferenced position of the ALS point, the "trajectory" vector is the georeferenced position of the origin of the IMU, the "scanner" vector is the laser ray in the scanner coordinate system, the "lever" vector is the location of the scanner's laser emission point in the IMU reference frame, $\boldsymbol{R}_{\text{bore}}$ is the 3×3 rotation matrix from the scanner frame to the IMU frame, and $\boldsymbol{R}_{\text{imu}}$ is the 3×3 rotation matrix from the IMU frame to the local level frame.

For an example of a geometric convention, consider the scanner vector in Equation 1. The scanner vector is a function of the lidar range and scanner angle measurements, but its computation requires knowledge of the scanner frame definition. Is the scanner frame a right-hand or left-hand system? If right-hand, what is the orientation? Does the x-axis point forward (direction of flight), y-axis to the right, and z-axis downward? Or does the x-axis point to the right, y-axis forward, and z-axis upward? Or is a different definition used? If the scanner is a traditional oscillating or rotating mirror type, is the scan angle defined as positive to the right or to the left of the vertical component of the scanner reference frame? These geometric conventions are required to correctly generate a scanner vector from the lidar range and scanner angle measurements that is compatible with the boresight rotation matrix, which, in turn, is defined by another set of geometric conventions.

Given an airborne lidar georeferencing equation that is mathematically consistent with the ALS measurements through knowledge of the ALS system geometric conventions, partial derivatives of the georeferencing equation with respect to each measurement can be formed for use with the General Law of Propagation of Variance (GLOPOV). For each set of measurements and corresponding uncertainties, GLOPOV is used to propagate the measurement uncertainties through the lidar georeferencing equation to generate a 3×3 covariance matrix that represents the uncertainty in the ground coordinate vector. Unfortunately, geospatial scientists are rarely supplied with the components necessary for per-point ALS TPU. Instead, they have access to point cloud data generated by proprietary software implementing a sensor-specific ALS georeferencing equation. Even if the necessary information were available, a non-trivial algorithm would need to be developed to generate the TPU.

To address—in part—this challenge, a plugin filter has been developed for the Point Data Abstraction Library (PDAL). This filter, called `filters.als_tpu`, generates per-point ALS TPU for a lidar point cloud under the following conditions:

1. Corresponding ALS trajectory information is available.
2. Information on the ALS system measurement uncertainties is available.
3. The ALS scanner uses an oscillating or rotating mirror as the scanning mechanism.

The ALS trajectory, which refers to a record of the position and attitude of the ALS IMU[1] at high temporal frequency, directly and indirectly provides many of the required measurements. Trajectories can be estimated from ALS point cloud data via a separate PDAL plugin filter, `filters.sritrajectory`, which should be publicly available in the near future. Measurement uncertainty information, however, must be obtained independently. It is usually sufficient to obtain the make and model of the ALS system IMU and laser scanner sensors; many of the required measurement uncertainties can be obtained from sensor datasheets found online. Refer to the documentation in the `filters.als_tpu` GitHub repository for the required format for the measurement uncertainties and an example of how the values were obtained for a particular ALS system. Note that the third condition—use of an oscillating or rotating mirror—eliminates ALS data generated with Risley prism scanners.

In order to generate ALS TPU for point clouds collected with different ALS systems, a generic set of geometric conventions are employed in `filters.als_tpu`. These conventions are not defined or explained in the source code, which renders the filter—at least partially—a black box. The purpose of this document is to remove the black box by documenting how the required measurements are obtained and the set of geometric conventions that were selected to create a generic lidar georeferencing equation. We also describe the GLOPOV implementation and the basic execution steps in the code.

[1] Trajectories are often solved to provide the position of the scanner origin (location of laser pulse emission), rather than the IMU origin, by incorporating the lever arm into the trajectory solution. As will be explained later in this document, we assume a zero length lever arm, which makes the IMU and scanner origins coincident.

2 Measurements

Fifteen measurements are used in the lidar georeferencing equation employed in `filters.als_tpu`:

1. Lidar range from the scanner laser emission location to the ground.
2. Scan angle - right/left.
3. Scan angle - forward/back.
4. X component of the IMU origin in the projected coordinate system of the ground points (trajectory x).
5. Y component of the IMU origin in the projected coordinate system of the ground points (trajectory y).
6. Z component of the IMU origin in the projected coordinate system of the ground points (trajectory z).
7. IMU roll with respect to the Local Level frame (trajectory roll).
8. IMU pitch with respect to the Local Level frame (trajectory pitch).
9. IMU heading with respect to the Local Level frame (trajectory heading).
10. Roll misalignment of the scanner frame with respect to the IMU frame (boresight roll).
11. Pitch misalignment of the scanner frame with respect to the IMU frame (boresight pitch).
12. Yaw misalignment of the scanner frame with respect to the IMU frame (boresight yaw).
13. X component of the laser emission location in the IMU frame (lever arm x)
14. Y component of the laser emission location in the IMU frame (lever arm y)
15. Z component of the laser emission location in the IMU frame (lever arm z)

Lidar range and the right/left and forward/back scan angles are inverted for each point using the ground point coordinates and IMU position (interpolated from the trajectory) at the time the ground point was measured. The inversion is detailed in section 5.

Trajectory x, y, z, roll, pitch, and heading are supplied by the trajectory data, with one exception: trajectory roll is always set to zero. When generating trajectory information from point clouds with `filters.sritrajectory`, roll is not recovered and is set to zero. For consistency, trajectory roll is therefore always set to zero by the filter, although this could be easily changed to accommodate trajectories with complete information. Sensitivity testing indicates that errors in trajectory roll have negligible impact, contributing at most a few percentage points of difference in TPU, and only when other significant trajectory errors exist.

Boresight roll, pitch, and yaw and lever arm x, y, and z are all set to zero in the filter. These values are generally fixed during a collection campaign and quite small in practice. Note that although these measurement values are set to zero, the corresponding measurement uncertainties are non-zero and contribute to the TPU solution. Using a zero value should have a negligible influence on the final TPU; however, this has not been confirmed with sensitivity tests.

Note that trajectory heading is assumed to have been corrected for wander angle. Note also that we use a heading value for the trajectory attitude but a yaw value for boresight alignment. Heading is traditionally used in trajectories and is defined as the clockwise (left-hand) angle about the local normal with respect to geodetic north, whereas yaw is a counter-clockwise (right-hand) angle about a z-axis with respect to a reference direction, typically the x- or y-axis.

3 Geometric Conventions

3.1 Reference Frames

1. IMU: x-axis forward, y-axis to the right, z-axis down; origin at IMU origin.
2. Scanner: x-axis forward, y-axis to the right, z-axis down; origin at location of emitted laser pulse.
3. Local Level: x-axis north, y-axis east, z-axis down; origin at ground point location.
4. Ground: Projected coordinate system and elevation datum of user choice, e.g., a UTM zone and ellipsoid height.
5. Trajectory: Same as the Ground frame.

The IMU and Scanner frames are relative to the direction of flight of an upright aircraft. The Local Level frame is often referred to as a north-east-down frame, or NED frame.

3.2 Rotations

Rotations about the x, y, and z axes are defined with active (rather than passive) rotation matrices with counter-clockwise positive (right-hand) angles:

$$\boldsymbol{R}_x(\omega) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega) & \text{-}\sin(\omega) \\ 0 & \sin(\omega) & \cos(\omega) \end{bmatrix} \tag{2}$$

$$\boldsymbol{R}_y(\phi) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ \text{-}\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \tag{3}$$

$$\boldsymbol{R}_z(\kappa) = \begin{bmatrix} \cos(\kappa) & \text{-}\sin(\kappa) & 0 \\ \sin(\kappa) & \cos(\kappa) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The rotation matrices $\boldsymbol{R}_{\text{imu}}$ and $\boldsymbol{R}_{\text{bore}}$ are composed from the individual rotation matrices above using an $\boldsymbol{R}_z\boldsymbol{R}_y\boldsymbol{R}_x$ multiplication order. $\boldsymbol{R}_{\text{imu}}$ is formed using the IMU roll (ω), pitch (ϕ), and heading (κ) angles provided in a trajectory solution; $\boldsymbol{R}_{\text{bore}}$ is formed using the boresight roll, pitch, and yaw angles.

3.3 Vectors

With reference to Equation 1:

1. Ground: The x, y, z location of a ground point in a projected coordinate reference system, e.g., a UTM zone (x, y) and ellipsoid height (z).
2. Trajectory: The x, y, z location of the IMU origin in the same projected coordinate system as the Ground vector.
3. Scanner: The laser ray vector in the Scanner frame.
4. Lever: The x, y, z location of the laser ray origin in the IMU frame.

The Ground and Lever vectors are self-explanatory. The Trajectory vector is also self-explanatory and the use of a projected coordinate system for this vector is discussed in the following section. The Scanner vector, however, is a function of several measurements:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{scanner}} = \boldsymbol{R}_x(-\theta_{RL})\boldsymbol{R}_y(\theta_{FB}) \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}, \tag{5}$$

where d is the Euclidean distance between the laser ray origin and the ground point, i.e., the magnitude of the laser ray, θ_{RL} is the right/left (+/− in sign) angle with respect to the direction of flight and the x-z plane in the Scanner frame, and θ_{FB} is the forward/back (+/− in sign) angle with respect to the direction of flight and the y-z plane in the Scanner frame. The right/left angle is what is typically referred to as an ALS scan angle; the forward/back scan angle is measured in the orthogonal direction and is only non-zero for oscillating or rotating mirror ALS scanners with a constant forward/back angle, e.g., channels 1 and 3 of an Optech Titan laser scanner. The inclusion of both right/left and forward/back scan angles enables us to add uncertainty from the laser beamwidth diameter in both the right/left and forward/back directions. Note that the right/left scan angle is negated in the $\boldsymbol{R}_x$ rotation matrix since a positive (rightward with respect to the direction of flight) scan angle requires a negative rotation about the forward-pointing x-axis of the Scanner frame.

3.4 Discussion

An IMU frame with the x-axis forward, y-axis right, and z-axis down produces a naturally ordered relationship with the IMU roll, pitch, and yaw angles: roll is about the x-axis, pitch is about the y-axis, and yaw is about the z-axis. By using active rotation matrices, we do not require the roll or pitch to be negated to be logical: aircraft nose up is positive pitch and aircraft bank to the right is positive roll. Since the IMU z-axis is downward, heading (loosely defined here as the rotation about the IMU z-axis as viewed from above) is equal to yaw, eliminating the ambiguity between these two directional definitions. Finally, by using an NED Local Level frame, heading (or yaw) angles logically connect the IMU frame to the Local Level frame; for example, when the aircraft is level with a north heading (heading = 0 degrees), the IMU and Local Level frames are parallel. However, since projected coordinate systems (used for the Ground and Trajectory frames) are defined with the x-axis parallel to grid east, y-axis parallel to grid north, and z-axis up, the basis of the Local Level frame must be changed from a north-east-down definition to east-north-up (ENU) before adding the right-most term in Equation 1 to the Trajectory vector.

The last sentence of the prior paragraph should cause concern, as it indicates a rigorous method for adding the two terms on the right side of Equation 1 is not being used. Prior to adding the laser vector in the Local Level frame to the Trajectory vector to compute the desired Ground vector, it is common to transform both vectors to an earth-centered earth-fixed (ECEF) system. After summing, the ECEF Ground vector can then be converted to latitude, longitude and ellipsoid height and ultimately projected to the ground coordinate system of choice. In our case, however, we add the laser vector in a Local Level frame directly to the Trajectory vector, which is in a projected coordinate system. This ignores rotational differences between Local Level and projected coordinates systems (e.g., meridian convergence) and scale differences.

This problem is addressed by using a quasi Local Level system. This is naturally the case when `filters.sritrajectory` is used to estimate the IMU trajectory position and attitude. The estimation makes direct use of the point cloud coordinate values without regard for the underlying coordinate reference system. Thus, the solved heading values are relative to projected grid north rather than geodetic north. Their use in the IMU rotation matrix, $\boldsymbol{R}_{\text{imu}}$, results in a laser vector that is transformed from the IMU frame to a quasi Local Level frame that is aligned with the projection grid. By using trajectory headings relative to grid north, the dominant rotational difference between Local Level and projected systems is eliminated. Projection scale is also contained in the estimated trajectory position coordinates and in the lidar ranges that are ultimately inverted from the trajectory and point cloud data. Therefore, for maximum consistency, it recommended to use a trajectory recovered from the point cloud rather than actual flightline trajectories.

3.5 Symbolic Realization

The above reference frame, rotation, and vector definitions are used to form the airborne lidar georeferencing equation used by `filters.als_tpu`. The equation can be generated in symbolic form with the Python script model.py in the GitHub repository. The symbolic variable names closely match the variable names used in the filter C++ code.

4 Variance Propagation

4.1 Review

Variance propagation is performed according to GLOPOV. Given a linear relationship between a set of measurements, $\boldsymbol{X}$, and a vector of computed quantities (e.g., a coordinate triplet), $\boldsymbol{Y}$, we can write:

$$\boldsymbol{Y} = \mathbf{A}\boldsymbol{X}, \tag{6}$$

where the linear relationship is contained in the $\mathbf{A}$ matrix. The covariance matrix $\boldsymbol{C}_Y$ of the computed quantities in $\boldsymbol{Y}$ is a linear combination of the covariance of the measurements, $\boldsymbol{C}_X$:

$$\boldsymbol{C}_Y = \mathbf{A}\boldsymbol{C}_X\mathbf{A}^{\mathrm{T}}. \tag{7}$$

For a nonlinear relationship between a set of measurements and a vector of computed quantities, a matrix of partial derivatives of the computed quantities with respect to the measurements serves as a linear approximation. This matrix of partial derivatives is often termed the "Jacobian" matrix, $\mathbf{J}$, and replaces the $\mathbf{A}$ matrix in Equation 7:

$$\boldsymbol{C}_Y = \mathbf{J}\boldsymbol{C}_X\mathbf{J}^{\mathrm{T}}. \tag{8}$$

Since the lidar georeferencing equation (Equation 1) is nonlinear, a Jacobian matrix of partial derivatives is required.

4.2 Implementation

4.2.1 Jacobian Matrix

Explicit partial derivatives of the lidar georeferencing equation were generated for each ground coordinate component (x, y, z) with respect to each measurement using the Python script model.py. The partial derivative expressions were copied to the C++ PDAL filter code where they are used to generate Jacobian matrices with three rows (one for each ground coordinate component) and fifteen columns (one for each measurement):

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial d} & \cdots & \frac{\partial x}{\partial LA_z} \\ \frac{\partial y}{\partial d} & \cdots & \frac{\partial y}{\partial LA_z} \\ \frac{\partial z}{\partial d} & \cdots & \frac{\partial z}{\partial LA_z} \end{bmatrix}, \tag{9}$$

where d is lidar range, LA_z is lever arm x, and $\ldots$ indicate partial derivatives of the remaining measurements. A Jacobian matrix is computed for each point based on the point's unique set of measurements.

4.2.2 Measurement Covariance Matrix

A measurement covariance matrix is generated for each lidar point. All measurements are assumed independent, leading to a diagonal matrix:

$$\boldsymbol{C}_X = \begin{bmatrix} \sigma_d^2 & & \\ & \ddots & \\ & & \sigma_{LA_z}^2 \end{bmatrix}. \tag{10}$$

The diagonal elements are generated by squaring the measurement standard deviations supplied by the user, with the following exceptions:

1. Lidar Range: If incidence angle is included in the TPU generation, the influence of beam divergence is added to the user supplied standard deviation as:

$$\sigma_d^2 = \sigma_{d_{user}}^2 + \left(d \tan(\alpha) \times \gamma/4\right)^2, \tag{11}$$

   where $\sigma_{d_{user}}$ is the user supplied lidar range standard deviation, d is lidar range, α is incidence angle, and γ is laser beam divergence. See Equation 8 in [Hartzell et al., 2015].

2. Right/Left Scan Angle: The influence of beam divergence is added to the user supplied standard deviation as:

$$\sigma_{\theta_{RL}}^2 = \sigma_{\theta_{RL_{user}}}^2 + \left(\gamma/4\right)^2, \tag{12}$$

   where $\sigma_{\theta_{RL_{user}}}^2$ is the user supplied scan angle standard deviation. See Equation 7 in [Hartzell et al., 2015].

3. Forward/Back Scan Angle: Contains only the the influence of beam divergence:

$$\sigma_{\theta_{FB}}^2 = \left(\gamma/4\right)^2. \tag{13}$$

   Note there is no user supplied standard deviation for the forward/back scan angle; it is assumed to be zero. Since the influence of the forward/back scan angle on a ground coordinate is almost perfectly correlated with boresight pitch, any uncertainty in the forward/back scan angle should be contained in the boresight pitch uncertainty. The forward/back scan angle is included in the model solely to capture the uncertainty imparted by beam divergence in the direction orthogonal to the right/left scan angle.

5 Filter Execution Steps

Instructions and examples for `filters.als_tpu` are found in the GitHub repository. The following summary steps through the computations performed by the filter to generate TPU for a single point. Much of this can be understood by examining the filter code.

1. Interpolate Trajectory Position and Attitude: Linearly interpolate the IMU x, y, z, pitch, and heading from the trajectory data using the point timestamp value.

2. Invert for Lidar Range: Compute the Euclidean distance of the laser vector from the interpolated IMU position to the point.

3. Estimate Incidence Angle: If selected for inclusion in the TPU computations, the laser ray to surface incidence angle is computed as:

$$\alpha = \arccos\left(-\frac{\vec{l}}{||\vec{l}||} \cdot \frac{\vec{n}}{||\vec{n}||}\right) \tag{14}$$

   where α is the incidence angle, $\vec{l}$ is the laser ray vector, and $\vec{n}$ is the surface normal vector estimated at the point location. Note that the surface normal vector must already exist, e.g., computed in a prior PDAL stage such as `filters.normal`, and be upward pointing. The negative sign flips the laser vector, $\vec{l}$, from downward to upward pointing.

4. Invert for the Right/Left and Forward/Back Scan Angles: Begin by transforming the laser vector computed in item #2 above from the ground reference frame to the scanner reference frame. With reference to Equation 1, since we assume a zero length lever arm and zero valued boresight angles, the lever arm vector is eliminated and $\boldsymbol{R}_{\text{bore}}$ is the identity matrix. We can also move the trajectory vector to the left side by subtracting it from the ground vector, which results in the laser vector in the ground reference frame. The revised version of Equation 1 is then:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{ground}} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{trajectory}} = \boldsymbol{R}_{\text{imu}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{scanner}} . \tag{15}$$

Multiply both sides by the inverse of $\boldsymbol{R}_{\text{imu}}$ to transform the the laser vector from the ground reference frame to the scanner frame:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{scanner}} = \boldsymbol{R}_{\text{imu}}^{-1} \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{ground}} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{trajectory}} \right) . \tag{16}$$

With the laser vector in the scanner frame obtained, we can use Equation 5 to extract the scan angles. Inserting the right/left and forward/back scan angles into $\boldsymbol{R}_x$ and $\boldsymbol{R}_y$ in Equation 5 and multiplying the terms on the right side of Equation 5, we get:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{scanner}} = \begin{bmatrix} d \sin(\theta_{FB}) \\ d \cos(\theta_{FB}) \sin(\theta_{RL}) \\ d \cos(\theta_{FB}) \cos(\theta_{RL}) \end{bmatrix} . \tag{17}$$

where d is lidar range, and θ_{FB} and θ_{RL} are the forward/back and right/left scan angles. The scan angles can then be extracted as:

$$\theta_{FB} = \arcsin\left(\frac{x}{d}\right) \tag{18}$$

and

$$\theta_{RL} = \arcsin\left(\frac{y}{d \cos(\theta_{FB})}\right) , \tag{19}$$

where x and y are from the scanner frame laser ray vector computed using Equation 16.

5. Build Measurement Covariance Matrix: Populate a diagonal matrix with elements per Section 4.2.2.

6. Propagate Measurement Covariance: Populate a Jacobian matrix of partial derivatives using the supplied, inverted, and assumed measurements. The partial derivatives are generated in symbolic form with the Python script model.py and organized as specified in Section 4.2.1. The order of the measurement partial derivatives matches the order of the measurement uncertainties along the diagonal in the measurement covariance matrix, which is the same as the order of the measurements listed in Section 2.

6 Summary Thoughts

This technical document is a first cut. It is unlikely that it is ideally organized, always clearly communicated, or perfectly describes the TPU methodology. The document should therefore be updated as errors, omissions, or ambiguities are identified and as changes to the TPU method are implemented in `filters.als_tpu`. This document was created with Overleaf, the online LaTeX editor—contact Craig Glennie or Preston Hartzell for collaborative access.

It is important to be aware of the limitations of `filters.als_tpu`:

1. The filter is only applicable to point cloud data generated with oscillating or rotating mirror scanners.
2. Point records must be timestamped using the same time standard as the trajectory.
3. If the option to include the influence of incidence angle in the TPU generation is selected, each point record must contain unit normal vector information. The normal vectors must point upward.
4. The point records should be sorted by time for efficient trajectory interpolation.
5. It will likely be advantageous to split the point cloud data by flightline. This enables a trajectory to be estimated for each flightline using `filters.sritrajectory`. However, if the point records are sorted by time and a single trajectory containing all flightlines in the point cloud is available, then splitting the data into flightlines may not be necessary.
6. Trajectory heading is assumed to be corrected for wander angle.
7. No temporal or spatial correlation is accommodated in the algorithm.

The last item is a significant deficiency and should make clear that while `filters.als_tpu` lowers the barrier for generating point cloud TPU, there is room for improvement.

References

[Hartzell et al., 2015] Hartzell, P. J., Gadomski, P. J., Glennie, C. L., Finnegan, D. C., and Deems, J. S. (2015). Rigorous error propagation for terrestrial laser scanning with application to snow volume uncertainty. *Journal of Glaciology*, 61(230):1147–1158.