A Better Way to Produce Canopy Models:

A Comparison of Three Different Process for the Creation of Canopy Height Model Products for Dense Forest Areas

Authors:

Pat Hall & Kylie Wagner

Contents

Abstract

The USFS relies on the in-house developed, free and open-source software called FUSION to produce CCMs and CHMs. FUSION is a powerful tool for processing and interacting with LiDAR data, but it requires that you use Windows BatchScript to process many LiDAR tiles at once. This makes FUSION very difficult to use from a data management and QA standpoint. This analysis is a cross-comparison of two additional models, one using the open-source application Python library PDAL and the other using Esri's ModelBuilder, to test the variance between all three of the tree canopy height results.

Background

There are a few different kinds of Forest Canopy Models (FCMs) that can be derived from LiDAR data. This paper will focus on one common type: Canopy Height Models (CMHs), which are made possible through the use of LiDAR technology. In the last 5 years, with the launch of the GEDI mission on the ISS, LiDAR has emerged as a pivotal tool in environmental monitoring and ecosystem analysis (Altaweel, 2023). LiDAR utilizes laser pulses to measure distances with high precision, providing detailed three-dimensional information about the Earth's surface (Liu et al., 2023). It has only been in the last few decades that LiDAR's application in forestry science has become particularly noteworthy. Previously, LiDAR was too expensive and covered areas too small to be useful. Fortunately for us today, LiDAR sensors can now be mounted to

relatively inexpensive drone platforms, and it is becoming increasingly common for high-resolution LiDAR sensors to be included on Satellite platforms.

---

fig.1

$$DSM - DTM = CHM$$

fig.1 shows the formula for calculating a Canopy Height Model. DSM = Digital Surface Model. DTM = Digital Terrain Model. The DSM is derived from LiDAR last returns, and is an interpolate representation of the ground elevation under the tree. The DTM is derived from LiDAR first returns, and is an interpolated representation of the highest points of the tree canopy. Subtracting the tree height (included in the DTM) from the elevation (included in the DSM) results in a CHM that represents only the height of the tree at a specific point. (Dang)

After CHMs have been produced for a specific forest area, we can use them to enhance our ability to analyze and monitor ecosystems comprehensively. By combining information on both vertical and horizontal canopy structures (Canopy Cover Models, or CCMs. Not covered in this paper), researchers gain a more holistic understanding of forest dynamics (Houghton, 2009). This integrated approach is particularly valuable for addressing ecological questions related to biodiversity conservation, habitat assessment, and ecosystem resilience, as well as resource inventory management (Houghton, 2009).

The production of CHMs is an involved process. There are numerous tools, both proprietary and FOSS, that can be used to create them. Unfortunately, as a given project scales in size, the task of managing LiDAR files, intermediary outputs, and final outputs becomes more arduous. The USFS relies on its own open-source tool, FUSION, developed by Bob McGaughey and the USFS Pacific Northwest Research

Station, for the creation of these Canopy Model Products. While FUSION is an impressive tool and yields accurate results, the learning curve for processing large forest areas comprised of multiple LiDAR files (a single forest area can be represented by thousands of individual LiDAR files) is steep.

FUSION relies on the use of the Windows command line to interact, which is one of the more difficult aspects of this process. Being that this tool is often intended to only be used on a single file at a time. To interact with bulk forest LiDAR products, the user must be familiar with Windows BatchScript, an archaic scripting language that's all but fallen out of use, beyond IT and Networking specialists. Due to FUSION's underlying code, it operates much faster when data is processed from one drive to a second drive, and multi-threaded processing, while possible, is difficult to implement. Setting up a large-scale FUSION CHM development project is time-consuming, makes data management difficult, and is computation resource inefficient. In this paper, we will be looking at two other processes as potential substitutions for the USFS' current standard process.

Objectives

The objective of this paper is to explore and identify an alternative process of CHM creation that is more efficient and accurate than FUSION. To accomplish this, we examined three distinct processes to achieve the same product; the creation of a CHM for Tuskegee National Forest in Alabama. Our first process establishes our Benchmark and utilizes USFS FUSION. Based on previous research, we know that FUSION

produces reliably accurate CHMs from LiDAR data (Shiota et al, 2017). The accuracy of the final product is the most important aspect of the process. Therefore, the other two processes developed for and tested in this paper will be measured on three criteria against how well FUSION performs in those areas. The two criteria we are measuring are speed of processing, and accuracy of the CHM as compared to FUSION.

Methods

This section is broken into four subsections; FUSION, ESRI ModelBuilder, PDAL + Rasterio, and Statistical Analysis. The first three subsections summarize step-by-step methods used for each of the CHM creation applications, while the last section is a summary of the methods used to determine the accuracy of their outputs. Each of the three CHM creation processes starts with LAZ format compressed point-cloud LiDAR data.

Process 1: FUSION

To create a CHM in FUSION, we utilized the Command Line toolkit, which is packaged with the FUSION LiDAR Visualization software. For the purposes of this paper, FUSION v 4.5 was used. The workflow is scripted using Windows BatchScript files. Windows BatchScript is executed via the command line and acts as a script of command-line functions executed sequentially.

Below is a step-by-step of the FUSION workflow we developed, with the functions utilized broken out. To view the exact parameters used for each function and

the Windows BatchScript files used, please refer to Appendix B. Fig. 2 and fig. 3 which shows the flowcharts depicting this process visually. A link to FUSION's Documentation can be found in references.

---

fig. 2

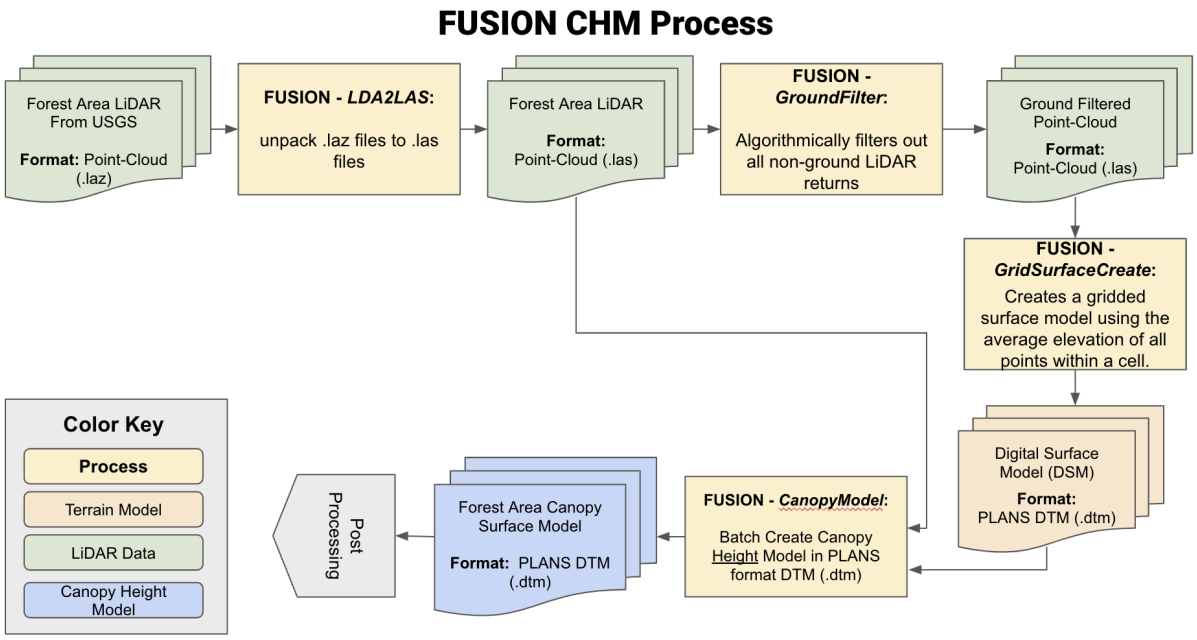

fig.2 shows a flowchart depicting the initial process in USFS FUSION For generating CHM files from LAZ LiDAR files.

01: Data Preparation + Canopy Modeling

For each LiDAR file processed, we use the point-cloud data to derive the DTM that will be used to create the CHMs. FUSION does not require a separate DSM to be created before processing into a CHM, as this is handled as a self-contained process in

the tool *Canopy Model.* The first BatchScript file performs the following processes for all available LAZ files in the container folder:

1. Uncompress LAZ file to LAS file.
    a. ***FUSION Tool - LDA2LAS***
2. Filter the LAS point-cloud file for ground elevation points.
    a. ***FUSION Tool - GroundFilter***
3. Derive DTM Ground Surface from Groundpoint LAS.
    a. ***FUSION Tool - GridSurfaceCreate***
4. Create CHM (see fig. 1)
    a. ***FUSION Tool - CanopyModel***
5. Delete the temporary LAS files created in steps one and two.

For speed, CanopyModel processes an LAZ from storage drive A to storage drive B. This script results in a series of CHM models stored as rasters in the USFS PLANS DTM format (.dtm).

02: Canopy Model File Format Conversion

Once all LiDAR files are processed into CHMs, they need to go through post-processing to convert them to the raster ASCII format (.asc). The ASCII format is a raster format that can be read into many GIS applications such as ESRI ArcGIS Pro or QGIS. In this step, all CHM files created in step 01 are exported to ASCII files with A 32-bit float pixel resolution. This step uses the FUSION tool *DTM2ASCII*. *DTM2ASCII* runs much faster when processing inputs from one storage drive and writing outputs to a second storage drive.

03: Quality Assurance

Once the Canopy Models have been converted to the ASCII raster format, they need to be checked for quality before moving on to the merging and cleaning steps of the process (step 04 and step 05). The easiest way we found to check for quality is to create a mosaic dataset in ArcGIS Pro and add the folder of ASCII rasters as the source data. Things to look out for when checking these files for quality:

- Correct Coordinate Systems
- Correct coordinates
- Model corruption
- Mismatched temporal resolutions (LiDAR tiles) - use older DTMs to patch holes in new rasters

04: Merge Canopy models

---

fig. 3

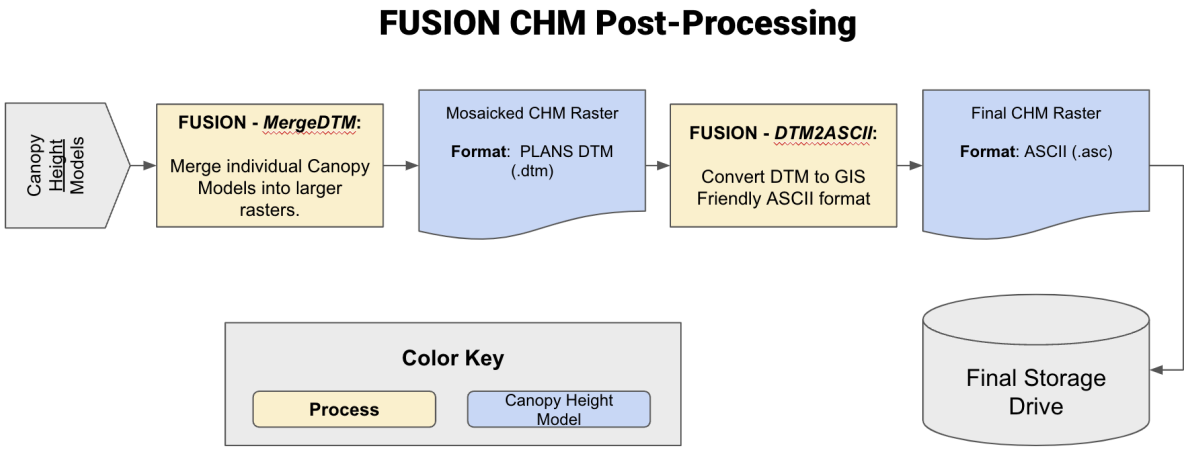

fig. 3 shows a flowchart depicting the post-processing steps taken in stages 02 and 03.

Now that LiDAR tiles have been processed into canopy models and checked for quality, the canopy model files need to be merged into a single file so that the forest area can be viewed and interacted with in a GIS application as a contiguous raster. FUSION's MergeRaster tool merges rasters in a specific order, with overlaps between files being overwritten by the next file in the order to be added so that only the most recent data is shown in the final canopy model product.

Similar to the *DTM2ASCII* conversion process described in Stage 02 and the *CanopyModel* process described in Stage 01, the canopy merging process runs faster if you write the merged product to a separate drive from where the original ASCII files are stored.

05: Clean Canopy Model Rasters

After the Canopy Models have been merged, they will likely represent an area that is slightly larger than the forest area we intend to look at. The final step in this process is to pull the newly created ASCII Canopy model files into a GIS application and use a shape file for the desired forest area extent to extract just the raster data that is relevant to our study area. This time, to do a final QA check on the canopy model product.

Process 2: ESRI ModelBuilder

This CHM process was accomplished by automating ArcGIS Pro geoprocesses in ArcGIS Pro ModelBuilder. Before creating a fully automated model that would process multiple LAZ files into a merged CHM file for an entire forest area, it was important to ensure that the base components of the model worked correctly; unpacking LAZ files, interpolating DSM and DTM files, and creating the CHM GTIFF. To check this, we used a published workflow from the PA Department of Conservation and Natural Resources, titled “LiDAR-derived Tree Canopy Heights using ArcGIS Pro". For the full ArcGIS Pro Model and submodels described in these next subsections, See Appendix C.

01: Unpack the LAZ files

LiDAR data from USGS 3DEP is stored in LAZ format point-cloud files, which are zip-formatted files for LiDAR data. Before we can utilize these files, they need to be unzipped into LAS files. Converting the LAZ file into an ArcGIS Pro compatible ESRI .lasd file is accomplished by utilizing the “Convert LAS" geoprocessing tool. The LAZ file is the input LiDAR tile that needs to be converted, while the target folder is the output folder where the unzipped ESRI .lasd files are stored. By choosing the “No Compression” option the tool knows to de-compress the input files, unzipping the LAZ files into the correct (.lasd) file format.

02: Filtering the LAS files

The next step of the process begins by adding an ESRI .lasd file into ArcGIS Pro. From the content pane, we filter the first returns from the LiDAR image by right-clicking on the “LAS Filters” tab and selecting the “1st Returns”. This produces the DSM for the process. Once the DSM has been filtered, the next step is to generate a raster of the first returns, done with the “LAS Dataset to Raster” geoprocessing tool in ArcGIS Pro. Specifying the interpolation type to “Binning”, cell assignment to “Maximum”, and void fill method to “Linear” per the instructions. We set the output data type to “32-bit Floating Point”. Additionally, set the output cell size and sampling value to “1”, which is consistent with the rest of the analysis.

From here, the same is done for the DTM, but instead of first returns, you filter the ground returns. Like the previous step, this is done by right-clicking on the “LAS Filters” tab and instead selecting the “Ground”. Again, this result will need to be transformed into a raster with the “LAS Dataset to Raster” geoprocessing tool. This time, specify the interpolation type to “Binning”, but instead set cell assignment to “Minimum”, and keep the void fill method to “Linear”. While keeping the output data type to “Floating Point” point classification follows the guidelines set out by the ASPRS. (LAS Specs, 2019)

03: Subtracting the DSM from DTM

At this point, both the DSM (first return) and DTM (ground) rasters have been created. From here, the next step is to subtract the DSM from the DTM by using the “Minus” geoprocessing tool (refer to fig.1 for the CHM formula). Assigning input raster 1

as the “first returns” raster and input raster 2 as the “ground returns”. This produces the initial CHM measured in meters. To remove any noise from the results, like negative numbers or outliers, use the “Con” geoprocessing tool, which intakes conditional functions to output a new raster. This tool removes any values under 0 and anything over 1.83 meters (6 feet; This is a height break commonly used by the USFS). This is the final CHM version, from here the next step is automating this process to run over a folder containing all the LAZ files for a forest area.

04: Model 1 - LAZ to LAS

---

fig. 4

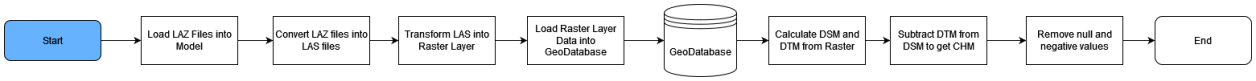

fig. 4 shows the process diagram that details each general step of Model 1, converting the .laz files within a folder into .lasd files, which are then filtered to produce DTM and DSM values. For the specific ModelBuilder flowchart see fig. 14 in Appendix C.

The first step was to unzip the LAZ file to LAS, just like the manual process. The main difference here was that the model needed to work over multiple files, rather than just the single input. Batching tools don't perform the same functions in ModelBuilder that they do in a map scene. So instead, to perform the same function, we added an “Iterate Files” module to the model. As seen in fig. 4, this model intakes a folder of LAZ files, iterates over them to name them, and moves them into a geodatabase, which is used to convert the LAZ files into ESRI .lasd files. ESRI .lasd files are transformed into a .lasd raster layer, which is then filtered to create the DTM and DSM. Once we have

the DSM and DTM rasters, we subtract the DSM from the DTM to create the CHM. This resulting CHM is run through the “Con” tool to remove the noise. With the full model run, the final product is a unique CHM raster for each of the files in the LAZ folder.

05: Model 2 - Rasters to Mosaic

---

fig. 5

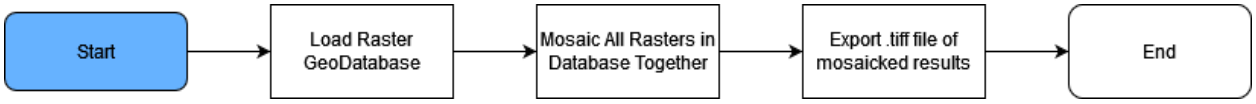

fig. 5 shows the process diagram that details each general step of Model 2, which mosaics all of the individual .lasd rasters into a single mosaic layer. For the specific ModelBuilder flowchart see fig. 15 in Appendix C.

With the first model complete, the next step is to combine the individual rasters into a collective mosaic of the whole AOI. Building off the results of Model 1, the raster database is the intake for the secondary “Iterate Rasters” module. This tool names the rasters within the database and outputs them into a separate dataset. This dataset is then input into a “Collect Values” tool, which works as a holding container to hold the output rasters until the iterator tool has finished running. Those “Output Values” are then run through the “Mosaic to new Raster” tool, which combines all of the raster into a single mosaic. This mosaic is output into a specified folder, in this case, “Finished_Mosaic”, where the TIFF file can be accessed.

06: Model 3 - All Together Now

fig. 6

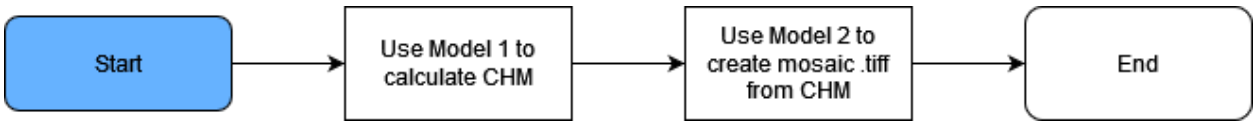

fig. 6 shows the process diagram that details each general step of Model 3, which combines the processes of Model 1 and Model 2 into one cohesive model. For the specific ModelBuilder flowchart see fig. 16 in Appendix C.

ModelBuilder restricts the number of iterators per model to only one. A limitation that is handled by implementing sub-models within a larger model. With our models, this meant creating an additional model as the “main” model. This “main” model is then populated with Model 1 and Model 2 as submodels from the project toolbox. This allows for both models to run as one continuous process. Just as with the previous models, the LAZ folder is the input to the first model and once that is complete, the second model can run. After both models have run to competition, the merged mosaic is output as a GTIFF to the specified folder and can be viewed within the ArcGIS Pro map scene.

Process 3: PDAL + Rasterio

The third process we developed utilizes PDAL and the Rasterio python library, organizing both in a Python script. PDAL is a collection of applications for processing and translating point-cloud data (Bell et al, 2024). PDAL allows the user to interact with

point cloud data in several ways, but primarily through the Windows command line. For this paper, we decided to execute PDAL pipeline processes, encoded as json files, as Python pipeline objects using the PDAL extension library for Python. This allowed us to keep our processes in a unified language (in this case python) and avoided having to process point-cloud files in a different application from our resulting raster files.

Rasterio is the other piece of this CHM creation process. Rasterio is a Python library that allows users to interact with rasters as numpy arrays. Rasterio is key to why we believe this process would be speedier than FUSION; Numpy arrays are exceptionally space efficient and handle arrayed mathematical operations much faster than other types of arrays. (*What Is NumPy*, 2022) The full code base can be viewed on Github in the repository m0sthuge / CanopyModel_PDAL. (Hall, 2024)

01: Compile Data + Interpolate DTM and DSM

The first step of our python script is to create DTM and DSM raster files from the LAZ point-cloud files that are available. To do this, we utilize PDAL. To instruct PDAL on the operations we want to perform for each LAZ file, we created two json “pipeline” files to describe the DTM and DSM creation process. Our initial pipelines were based on work done by Simon Planzer, and PDAL’s pipeline documentation (Planzer, 2020; Bell et al, 2024). With the pipelines created, we wrote a Python function to locate all LiDAR files within a specified folder and compile them as a list of file paths. We then used a for loop to cycle all the LAZ files through the DTM and DSM pipelines. An example of a PDAL pipeline written as a declarative json can be seen in fig. 7.

fig. 7

```
[
    {
        "type":"readers.las",
        "filename":"Test.laz",
        "compression":"laszip"
    },
    {
        "type":"filters.expression",
        "expression":"Classification == 1"
    },
    {
        "type":"writers.gdal",
        "filename":"test_dsm.tif",
        "output_type":"idw",
        "gdaldriver":"GTiff",
        "resolution": 1,
        "radius": 1,
        "nodata": -10
    }
]
```

fig. 7 shows the pipeline file DSM.json. The first section reads in the specified LAZ file, the second section filters it for "nonground" points, and the third section writes out the DSM file as a GeoTIFF (.tif) using inverse distance weighting to interpolate pixel values.

The order of operations is as follows:

1. Identify and compile all LAZ files in a specified folder
2. Create a DTM from GTIFF from each LAZ file
3. Create a DSM GTIFF from each LAZ file

This outputs the DTM and DSM rasters for the LiDAR file in the forest area that will be used in the next stage of the process. All output rasters use -10 as the placeholder for null values.

02: Canopy Height Model Creation

Referring back to fig. 1, remember that a CHM can be boiled down to DSM - DTM = CHM. For this stage, the Rasterio Python library is utilized. With Rasterio, both the newly created DTM and the newly created DSM are loaded into numpy arrays. Because we use two different processes to interpolate the DTM and DSM, the rasters can vary in size by a column or two. Unfortunately, numpy array mathematical operations only work when the two arrays share the same shape (shape in numpy refers to the array's dimensions, i.e. number of rows and number of columns) to account for this, we created a function to check the shape of each DTM and DSM array loaded into memory; if arrays are not the same shape, the smaller array is padded with blank columns or rows. This should not affect the output of the final CHM, as null values near the edges of the raster datasets will get overwritten or masked when rasters are mosaiced and extracted later on in the process.

Once the DTM and DSM files are loaded into arrays, and the array shapes are checked for consistency, we can perform the mathematical operation of subtracting them to arrive at a CHM. In Python, this looks like:

```
raster_chm = raster_dsm - raster_dtm
```

The resulting variable raster_chm is then written out to a uniquely named GeoTIFF. Stage 01 and Stage 02 loop using a for loop, and the result is a CHM GeoTIFF file for every LAZ file input. This process was based on PyGIS.io documentation, a free textbook designed for using Python for open-source spatial programming and remote sensing (Mann et al, 2023).

03: Mosaic Canopy Height Models

The final step of the Python script is to mosaic all the produced CHMs into a single contiguous CHM for the area they have been produced. This stage utilizes Rasterio's merge function and is based on classwork produced by the publicly owned and operated company CSC Finland - IT Center for Science, and based on coursework from the University of Helsinki (Tenkanen, 2018). We use Python to create a list of all the CHMs made in Stages 01 and 02. All the CHMs are then written into numpy arrays with Rasterio, and then those arrays are mosaicked into a single master array based on the georeferencing information included in their file headers. The new array is then written to a new GTIFF file which is a contiguous raster dataset for the area that the CHMs cover.

Statistical Analysis

We used a number of methods to compare the CHM rasters produced by the 3 processes previously mentioned. As previously stated in this paper, for the purposes of this analysis we are assuming FUSION's process to be most accurate and are therefore using it as our Benchmark for Processes 2 and 3 (Shiota et al, 2017). Thus, all measurements and statistical tests are done as relative comparisons against FUSION. We broke our Analysis into 2 parts, processing time and accuracy. For Processing Time, we started the timer from the moment the process began and stopped it at the moment it ended. For processes 2 and 1, we used the built-in timestamps printed throughout

processing; for process 3, we used a manual timer. Times were directly compared to each other, and the difference in timing was recorded.

To compare the accuracy of the resulting CHMs against the Benchmark FUSION CHM, we used several metrics and tests; Mean Error, Paired T-Test, and a Wilcox Signed-Rank Test. To find Mean Error, we used the raster calculator function in ArcGIS Pro to create rasters showing the absolute difference between the CHMs produced by Processes 2 (ESRI ModelBuilder) and 3 (PDAL + Rasterio) and the CHM produced by Process 1 (FUSION). We then used the statistics summary tool to find the mean of the error raster values.

The data that we are comparing in this study is mostly normal, but all three result CHMs have a negative skew to their data distributions. However, this negative skew is balanced by the size of the datasets; the final CHM produced in each process contains approximately 2.25 million data points. Because of this, we elected to use both a parametric test and a non-parametric test for determining the experimental processes' similarity to the Benchmark process.

The parametric test we chose is the Paired T-test. This test is commonly used for determining statistically significant differences in datasets measured from the same subject at two different points in time (Bangdiwala, 2013). Our purposes are slightly different as we're measuring datasets collected at the same time, but produced through different processes. We believed this was a similar enough use-case for the Paired T-Test to produce a result that would be informative of the accuracy of our experimental processes to the results of the Benchmark process. Additionally, this test is known to be

effective with non-parametric data when the sample sizes are large enough (> 30 samples) (Bangdiwala, 2013). The Paired T-test outputs a T-statistic which shows the scale of difference between the mean values of the experimental CHM and the Benchmark CHM, and the p-value which signifies whether or not the difference is statistically significant (our alpha threshold is .05).

As a secondary test, we chose to perform a Wilcox Signed-Rank Test, a non-parametric test designed to find the similarity between two measurements of the same subject. The Wilcox Signed-Rank Test tells us whether or not the CHM products have a statistically significant difference, but does not give an indicator of how different like the T-test does.

The Paired T-Test and Wilcox Signed-Rank test were both performed in a Python script (viewable in Appendix D) utilizing the Rasterio library to open the CHM rasters as numpy arrays, then performing the tests using the precompiled functions found in the SciPy Python library (SciPy Documentation, 2024).

Results

The objective of this paper is to find alternative approaches to FUSION when it comes to the creation of CHM products from LiDAR point-cloud datasets. As mentioned previously, we know that FUSION produces a high-quality CHM. Therefore, our results are a series of analyses that use the FUSION-produced CHM of Tuskegee National Forest as a Benchmark for comparison.

As described in the Statistical Analysis portion of Methods, our results can be divided into three categories: the Difference in Processing time between Process 1 and Processes 2 and 3, the Mean Error in the final CHM product from Processes 2 and 3 from the output of Process 1, and the Statistical similarity of the final CHMs produced by Processes 2 and 3 against Process 1. These results can be viewed below in tables 1-4.

---

table 1

CHM Processing Times		
Process	**Time (mm:ss)**	**∆ From Benchmark (mm:ss)**
Process 1 (Benchmark)	55:37	00:00
Process 2 (ESRI Model Builder)	31:00	24:37
Process 3 (PDAL + Rasterio)	04:10	51:27

Table 1 provides a comprehensive comparison of the CHM processing times, starting from the moment the process was initiated and ending at the moment the mosaicked CHM raster image stored as a GTIFF was produced. This comparison highlights the efficiency of each process.

table 2

Mean Error in CHMs as Compared Against Benchmark		
Process	**Mean Error**	**SD of error**
Process 2 (ESRI Model Builder)	-5.34 meters	± 6.95 meters
Process 3 (PDAL + Rasterio)	-10.34 meters	± 6.91 meters

Table 2 shows the pixel-by-pixel Mean Error and Standard Deviation of the error between the CHM rasters produced by processes 2 and 3 and the CHM raster produced by the Benchmark, process 1 (FUSION).

table 3

Similarity to Benchmark: Paired T-Test		
Process	**T-Statistic**	**P-Value**
Process 2 (ESRI Model Builder)	277.77	> .001
Process 3 (PDAL + Rasterio)	2757.73	> .001

Table 3 shows the results of two paired T-tests performed first on the process 2 CHM and the process 1 CHM (Benchmark) and then on the process 3 CHM with the process 1 CHM (Benchmark). The Null hypothesis in both instances was that the Experimental CHM would be the same as the Benchmark CHM. The T-statistic is a measure of how many standard errors the mean difference is away from 0.

table 4

Similarity to Benchmark: Wilcox Signed-Rank Test	
Process	**P-Value**
Process 2 (ESRI Model Builder)	> .001
Process 3 (PDAL + Rasterio)	> .001

Table 4 shows the results of two Wilcox Signed Rank Tests performed first on the process 2 CHM and the process 1 CHM (Benchmark) and then on the process 3 CHM with the process 1 CHM (Benchmark). The null hypothesis in both instances is that the Experimental CHM would be the same as the Benchmark CHM.

fig. 8

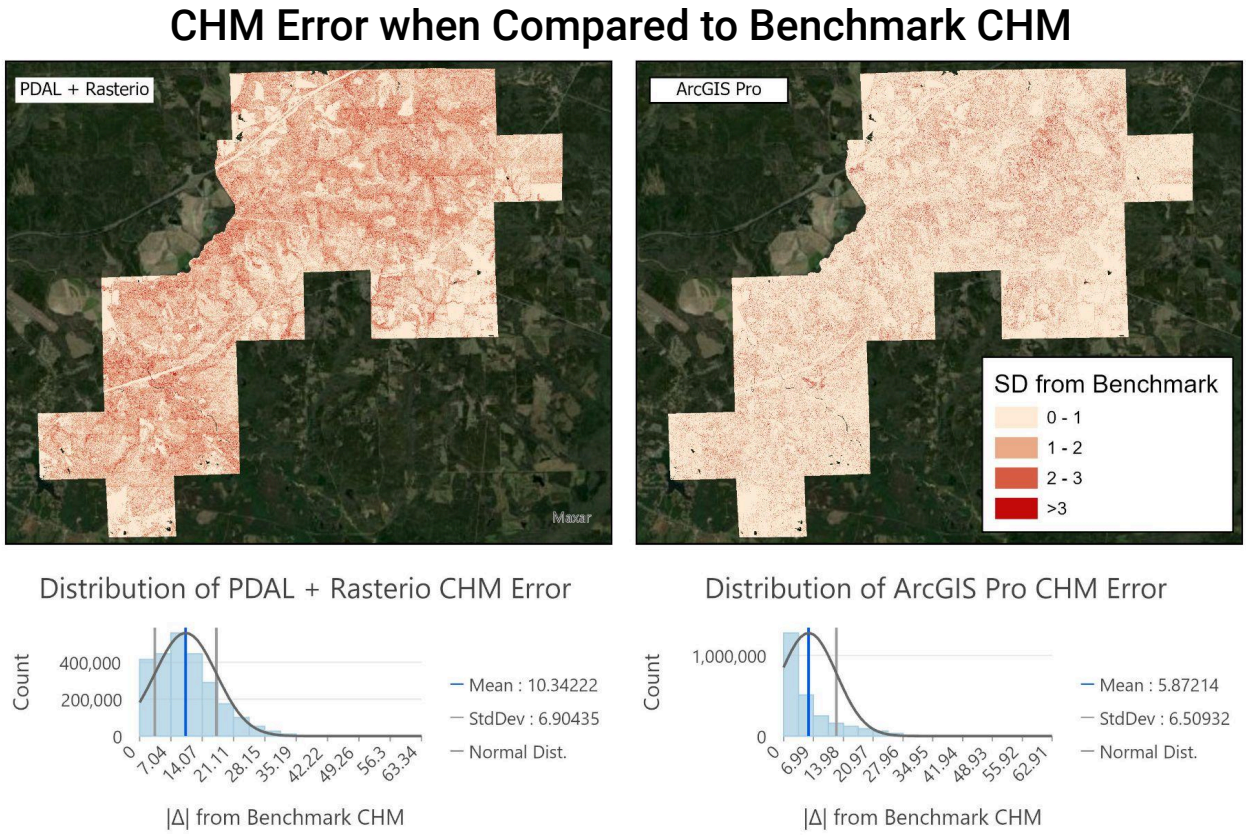

fig. 8 shows two rasters of Tuskegee National Forest, AL where the pixel value is set to the absolute difference between the experimental process produced CHM and the Benchmark CHM. Both rasters are symbolized such that pixels are darker red the more standard deviations they are from the Benchmark.

fig. 9

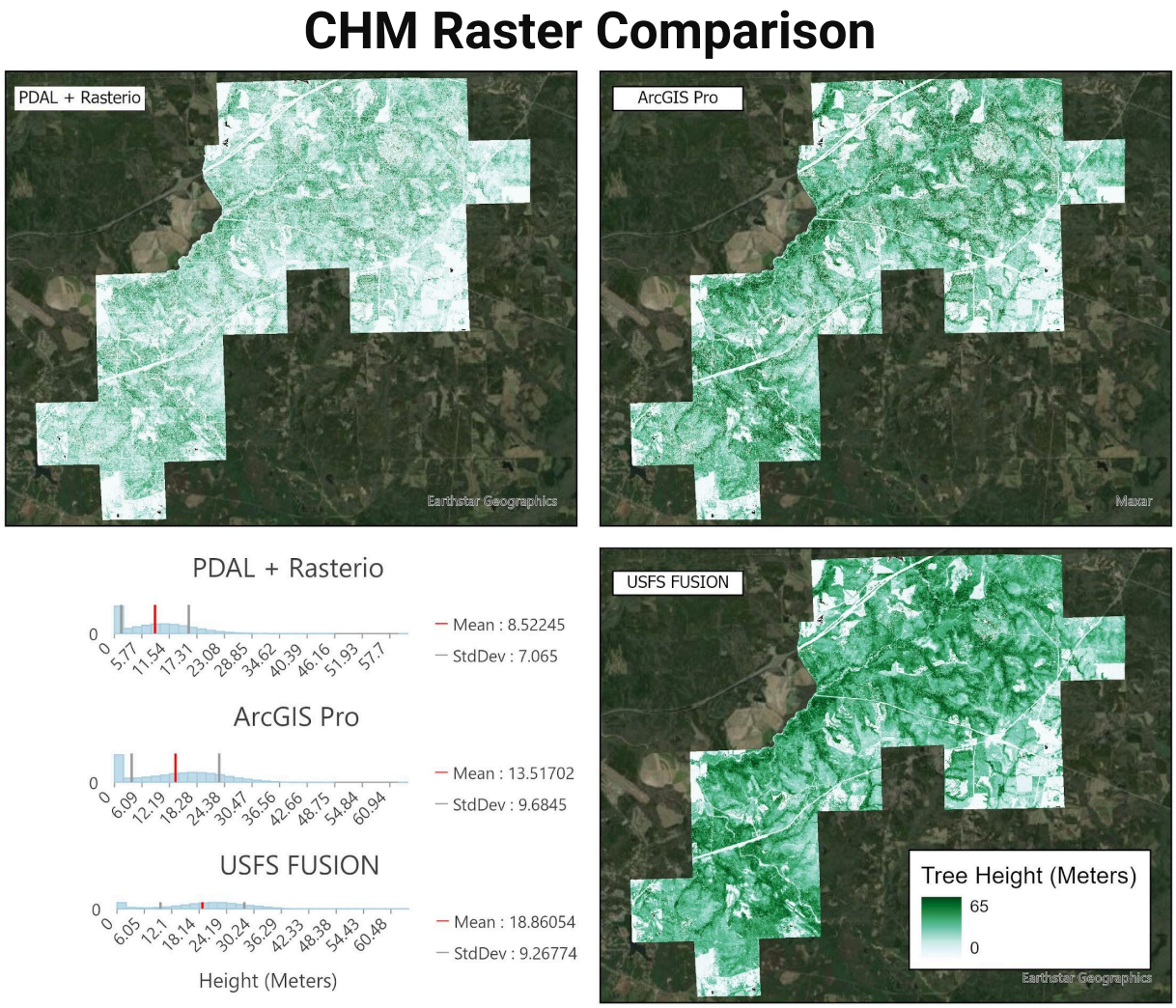

fig 9 shows the outputs from the three processes described in the methods. The lower left corner shows histograms and key summary statistics for each raster dataset. FUSION produces a CHM with the highest average canopy elevations, ESRI Model Builder produces results with the second highest average canopy elevations, and PDAL + Rasterio produces canopy height significantly lower than both.

Discussion

The objective of this study was to compare two different processes for creating CHM products for dense forest areas against a Benchmark process. As seen in the methods and results sections, we tested for speed and accuracy. As a reminder, the two experimental processes that we compared to the FUSION Benchmark process were processes that utilized ESRI ModelBuilder and a Python method utilizing PDAL + Rasterio.

The results of our study show that while all three processes can produce CHMs from the same source point-cloud files, there were significant variations in the accuracy of the CHMs produced by each process, and the accuracy of the resulting CHM product when compared to the FUSION Benchmark CHM. From our results, we learned a few things. First, we learned that it is possible to create CHMs with processes that do not rely on FUSION. Both of our experimental processes were able to produce a CHM. Second, we learned that our experimental CHM processes are much faster at producing a CHM than the FUSION process is. In table 1, you can see that process 2 performed the “CHM creation process” and produced CHMs 24 minutes and 37 seconds faster than the FUSION Benchmark process, and process 3 performed a staggering 51 minutes and 27 seconds faster than the Benchmark. Third, we learned that our experimental processes ultimately did not produce CHMs accurate to the Benchmark FUSION CHM.

In table 2, the “Mean Error” column shows us that process 2 produces canopy height values that are on average 5.34 meters lower than the corresponding FUSION CHM values. Process 3 shows an even greater degree of error from the FUSION CHM, with values lower by 10.34 meters on average. A visual representation of the pixel-by-pixel error can be seen showing how many standard deviations away from the Benchmark each pixel’s value is in fig. 8. The more red a pixel is, the less accurate it is to the FUSION model. However, looking at the error rasters isn’t necessary to understand how different the experimental process CHMs turned out from the FUSION process; the difference is visible in fig. 9 which shows a side-by-side comparison of the CHM rasters created by each process. Both the experimental processes produce CHMs with lower height values than what is created by FUSION.

These inaccuracies are reinforced by the two statistical tests we chose to perform testing the similarity of the experimental process CHMs to the FUSION CHM. The paired T-tests we performed showed returned p-scores far below our threshold of .05. This indicated we should reject our null hypothesis that the experimental processes produced CHMs identical to the CHM produced by FUSION. Both processes returned high T-statistics, as seen in table 3, a further indicator of the high variability of the experimental process CHMs from the Benchmark FUSION CHM. For further reinforcement, we also chose to include a non-parametric test for similarity in the form of a Wilcox Signed-Rank test, see table 4. This test only returns a p-score, but the sub .05 p-scores returned point to our experimental process CHMs being significantly different from the FUSION-produced CHM.

Why are the experimental process CHMs so different from the FUSION produced CHM? We believe the issue may lie in the first step of all three processes; interpolation of DTM and DSM rasters. In the creation of our experimental processes, we spent most of our focus on creating alternative executable processes capable of producing a CHM raster. When it came to filtering “ground” and “not-ground” LiDAR points from the initial LAZ files, we chose to use the points as they were already classified: “2 = “ground”, “1” = “not-ground”. (LAS Specs, 2019). FUSION on the other hand, has an algorithmic ground-point filtering process it uses before DTM is created (FUSION - GroundFilter in the flowchart shown in fig. 2). This very likely produces different ground heights that would affect the CHM calculation further on in the process. (McGaughey, 2023) Future iterations of this project could include a more nuanced approach to ground-point filtering in the experimental processes to improve accuracy. A good starting point for this would be to look into the source code for the GroundFilter process and attempt to replicate it in the experimental processes.

Another area that could impact accuracy, specifically in Process 3, is the use of NumPy arrays for raster calculations instead of performing the calculations on the raw rasters. The CHM produced by Process 3 is noticeably less accurate than Process 2, and a reason for this could be a loss of data in converting raster values to a numpy array. A deeper look into how data values are stored in numpy arrays could help improve this process in the future.

Conclusion

Our study highlights the strengths and weaknesses of each process. FUSION remains the most accurate process for producing CHMs, but it continues to be difficult to use from a data management and quality assurance standpoint. ESRI ModelBuilder is easier to use than FUSION, but it is still time-consuming to run and produces results less accurate than FUSION. PDAL + Rasterio is the fastest of the three processes, but it produces CHMs with the highest rate of errors.

In conclusion, our study provides a starting point for alternative CHM creation processes. While FUSION remains the best option for creating CHMs from LiDAR datasets due to its output accuracy, our experimental processes showed promise that they could eventually be as accurate as FUSION.

References

Altaweel, Mark. “GEDI: A Tool for Forest Carbon Maps.” *Geography Realm*, 5 Dec. 2023, https://www.geographyrealm.com/gedi-tool-forest-carbon-maps/

Band Math w. Rasterio—Python Open Source Spatial Programming & Remote Sensing. (n.d.). Retrieved April 22, 2024, from https://pygis.io/docs/e_raster_math.html

Bangdiwala, S. I. (2013). Before–after paired data. *International Journal of Injury Control and Safety Promotion*, *20*(3), 302–304. https://doi.org/10.1080/17457300.2013.823017

Bell, A., Chambers, B., Butler, H., & Others. (2024). About—Pdal.io [Documentation]. Pdal.Io. https://pdal.io/en/2.7-maintenance/about.html

Bell, A., Chambers, B., Butler, H., & Others. (2024). Pipeline—pdal.io. pdal.Io. https://pdal.io/en/2.7-maintenance/pipeline.html

Dang, Nguyet. *Error Propagation in Carbon Estimation Using The Combination of Airborne LiDAR and Very High Resolution GEO-EYE Imagery in Ludhikhola Watershed, Nepal*. 2012. *ResearchGate*, https://doi.org/10.13140/RG.2.2.20020.65920

Hall, P. (2024). CanopyModel_PDAL [Code Repository]. GitHub. https://github.com/m0sthuge/CanopyModel_PDAL

Houghton, R. A., et al. “Importance of Biomass in the Global Carbon Cycle.” *Journal of Geophysical Research: Biogeosciences*, vol. 114, no. G2, 2009. *Wiley Online Library*, https://doi.org/10.1029/2009JG000935

LAS Specification 1.4- R14. (2019). The American Society for Photogrammetry & Remote Sensing. https://www.asprs.org/wp-content/uploads/2019/03/LAS_1_4_r14.pdf

LIDAR-derived tree canopy heights using arcgis pro retrieved April, 23, 2024 from *https://www.srbc.gov/pennsylvania-elevation-working-group/docs/lidar-canopy-heights-arcgispro.pdf,*

Liu, X., Zhang, L., Zhai, X., Li, L., Zhou, Q., Chen, X., & Li, X. (2023). Polarization Lidar: Principles and Applications. *Photonics*, *10*(10), Article 10. https://doi.org/10.3390/photonics10101118

Mann, M., Chao, S., Graesser, J., & Feldman, N. (n.d.). (2023) PyGIS - Python Open Source
Spatial Programming & Remote Sensing [Textbook]. PyGIS.Io. Retrieved May 7, 2024, from https://pygis.io/docs/a_intro.html#

McGaughey, R. J. (2023). *FUSION/LDV: Software for LIDAR Data Analysis and Visualization*. United States Department of Agriculture. http://forsys.sefs.uw.edu/software/fusion/FUSION_manual.pdf

Planzer, S. (2020, May 7). *Producing a Canopy Height Model (CHM) from LiDAR* [Portfolio]. Simonplanzer.Com. https://www.simonplanzer.com/articles/lidar-chm/

SciPy User Guide—SciPy v1.13.0 Manual. (2024). [Documentation]. SciPy User Guide. https://docs.scipy.org/doc/scipy/tutorial/index.html

Shiota, H., Tanaka, K., & Nagashima, K. (2017). LiDAR Data Analysis with Fusion/LDV for
Individual Tree Measurement. *Journal of Biodiversity Management & Forestry*, *2017*. https://doi.org/10.4172/2327-4417.1000184

Tenkanen, H. (2018). Creating a raster mosaic—Intro to Python GIS documentation [Education Course]. Intro to Python GIS. https://automating-gis-processes.github.io/CSC18/lessons/L6/raster-mosaic.html

What is NumPy? —NumPy v1.26 Manual. (2022). Numpy.Org. https://numpy.org/doc/stable/user/whatisnumpy.html

Appendix A
Meta Data

For this paper, the LiDAR data was acquired from the United States Geological Survey (USGS) National Map, produced by the USGS Three Dimensional Elevation Program (3DEP). LiDAR data is for the area covering Tuskegee National Forest, which contains 42 total tiles. All LiDAR data used in this project was imaged with:

- 2017 temporal resolution
- 1-meter spatial resolution
- UTM zone 16N
- NAD 1988

All files were downloaded in the compressed .laz point-cloud format. The initial acquisition of 3DEP Lidar files began in late August 2023, and was finished by early October 2023. Files to download were collected using USFS shapefiles to select relevant LiDAR Files on the national map. The LiDAR file names were then collected in a text file, and the torrenting application uGet (https://apps.nationalmap.gov/uget-instructions/) was used to bulk download the relevant files, which is the USGS recommended method for doing so.

Appendix B

FUSION BatchScript Examples

fig. 10

FUSION Stage 01 - Data Preparation + Canopy Modeling

```
Filename: Process.bat

@echo off

REM add Fusion commandline executables to the System Path.
REM Instruct FUSION to use the 64bit version of commandline tools (speedier
REM and more efficient for 64-bit workstations)
Path %PATH%;C:\FUSION
set FUSION64=TRUE
set LTKLOG=canopy_model.log

REM Prompt user for Forest Folder name to process
set /p NatForest=Enter Folder name for Forest:

REM Prompt user for UTM Zone input
set /p UTMZone=Enter UTM Zone:

REM Change Directory to the national forest specified
cd %NatForest%

REM Create filelist.txt with .laz files to process
DIR /b *.laz > filelist.txt
echo Compiled a list of .laz files to process in %NatForest%

REM Create output subdirectories for surface models, temp LAS files, and
output canopy Models

if not exist surface_model mkdir "surface_model"
if not exist temp_las mkdir "temp_las"
if not exist temp_ground_las mkdir "temp_ground_las"
if not exist output_canopy_cover mkdir "output_canopy_cover"
if not exist output_canopy_height mkdir "output_canopy_height"
```

```
echo "temp_ground_las" created in %NatForest%
echo "surface_model" created in %NatForest%
echo "temp_las" created in %NatForest%
echo "output_canopy_cover" created in %NatForest%.
echo "output_canopy_height" created in %NatForest%.

REM For loop that runs through all files compiled in filelist.
      for /F "eol=; tokens=1* delims=,. " %%i in (filelist.txt) do (

            REM unpack LAZ to LAS and Derive DEMs
            LDA2LAS /verbose /cleanlas %%i.laz temp_las\%%i.las

            GroundFilter temp_ground_las\temp_GroundPoints.las 5
            temp_las\%%i.las

            GridSurfaceCreate surface_model\%%i_surface.dtm 1 m m 1
            %UTMZone% 2 2 temp_ground_las\temp_GroundPoints.las

            REM Create Canopy Cover Model & Canopy Height Model

            Cover surface_model\%%i_surface.dtm
            output_canopy_cover\%%i_canopy_cover.dtm 1.83 10 m m 1
            %UTMZone% 2 2 temp_las\%%i.las

            CanopyModel /nofill /ground:surface_model\%%i_surface.dtm
            output_canopy_height\%%i_chm.dtm 5 m m 1 %UTMZone% 2 2
            temp_las\%%i.las

            REM Delete temporary LAS files
            DEL temp_ground_las\temp_GroundPoints.las
            DEL temp_las\%%i.las
      )



echo %NatForest% CCMs and CHMs processed.
set LTKLOG=
```

fig. 10 shows the Windows BatchScript used to process canopy models with the USFS FUSION-LTK *command line toolkit.*

fig. 11

FUSION Stage 02 - CHM File Format Conversion

```
Filename: Bulk_dtm2ascii
@echo off

REM add Fusion commandline executables to the System Path.
REM Instruct FUSION to use the 64bit version of commandline tools (speedier
REM and more efficient for 64-bit workstations)
Path %PATH%;C:\FUSION
set FUSION64=TRUE
set LTKLOG=dtm2ascii.log

Set /p selection=Converting CHMs (1) or CCMs (2)?

If %selection% == “1” (

      REM set National Forest name. Needs to be same as Forest folder
      containing REM ASCII files being converted.
      Set /p NatForest=Enter Folder name for Forest:

      REM set state folder to navigate to.
      Set /p State=Enter Folder name for state:

      REM set the output directory for ascii files. Located on different
      drive to REM optimize processing speed.
      set output=D:\CanopyHeightModels

      REM navigates to the output directory on another drive.
      cd /d %output%

      REM Creates a folder in the output directory to write output ASCII
      Canopy REM Models to.
      if not exist %NatForest% mkdir %NatForest%

      REM Change directory back to Directory storing Canopy Models in DTM
      format
      cd /d
```

```
      F:\FUSION\CanopySurface\%state%\%NatForest%\output_canopy_height

      REM Create filelist.txt with .laz files to process
      DIR /b *.dtm > dtm2ascii_filelist.txt
      echo Compiled a list of .dtm files to process in %NatForest%

      REM convert all DTMs in filelist to ASCII files
            for /F "eol=; tokens=1* delims=,. " %%i in
            (dtm2ascii_filelist.txt) do (
                        DTM2ASCII /raster %%i.DTM
                        %output%\%NatForest%\%%i.asc
                        )

      echo %NatForest% processed.
      set LTKLOG=
      )

If %selection% == "2" (
      REM set National Forest name. Needs to be same as Forest folder
      containing REM ASCII files being converted.
      Set /p NatForest=Enter Folder name for Forest:

      REM set state folder to navigate to.
      Set /p State=Enter Folder name for state:

      REM set the output directory for ascii files. Located on different
      drive to REM optimize processing speed.
      set output=D:\CanopyCoverModels

      REM navigates to the output directory on another drive.
      cd /d %output%

      REM Creates a folder in the output directory to write output ASCII
      Canopy REM Models to.
      if not exist %NatForest% mkdir %NatForest%

      REM Change directory back to Directory storing Canopy Models in DTM
      format
      cd /d F:\FUSION\CanopySurface\%state%\%NatForest%\output_canopy_cover
```

```
REM Create filelist.txt with .laz files to process
DIR /b *.dtm > dtm2ascii_filelist.txt
echo Compiled a list of .dtm files to process in %NatForest%

REM convert all DTMs in filelist to ASCII files
      for /F "eol=; tokens=1* delims=,. " %%i in
      (dtm2ascii_filelist.txt) do (
                  DTM2ASCII /raster %%i.DTM
                  %output%\%NatForest%\%%i.asc
                  )
echo %NatForest% processed.
set LTKLOG=
)
```

fig 11. Shows the bulk_dtm2ascii.bat *script used by the GAMLab to convert Canopy Models from the PLANS DTM format (.dtm) to the ASCII raster format (.asc).*

fig. 12

FUSION Stage 03 - Quality Assurance

```
Filename: create_model_list.bat

@echo off
setlocal enabledelayedexpansion

rem Set the directory path
set /p NationalForest=Enter name of folder holding Canopy Model Files:

rem Set the output file name
set /p year=Enter the year of files you want to collate:
set output=filelist_%year%.txt

rem Navigate to the directory holding National Forest canopy rasters
cd %NationalForest%

rem Initialize the output file
echo. > %output%
```

```
rem Loop through files in the directory
for %%F in (%NationalForest%\*%year%*.asc) do (

rem Append filenames to the output file
    echo %%~nxF >> %output%
)

echo File list generated: %output%

endlocal
```

fig. 12 shows an example script for collating a list of files to merge in stage 04. This is useful when the bulk acquisition from the National Map includes LAZ files from multiple years for the same space. This script will select out only files from the specified year, assuming file names follow the USGS standard file naming conventions.

fig. 13

FUSION Stage 04 - Merge Canopy Models

```
Filename: merge_ascii_patch.bat
@echo off

REM add Fusion commandline executables to the System Path.
REM Instruct FUSION to use the 64bit version of commandline tools
(speedier REM and more efficient for 64-bit workstations)
Path %PATH%;C:\FUSION
set FUSION64=TRUE
set LTKLOG=merge_ascii.log


REM set output folder for NatForest. needs to match the name of the
existing REM output folder in the output directory.
set /p NatForest=Enter Folder name for Forest:
set /p outputName=Enter output file name:

REM allows user to enter the file list names to be used in the raster
merge.
set /p FileList=Enter name of .txt file that contains files to be merged:
set /p FileList_patch=Enter name of .txt file that contains newest files
```

```
to patch older areas.

set output=G:\FinalCHM

cd %NatForest%

MergeRaster /verbose /overlap:new %output%\%outputName%.asc %FileList%
%FileList_patch%

echo %NatForest% processed.
set LTKLOG=
```

Fig. 13 shows an example script that can be used to merge Canopy models where we have both older files and newer files that need to be represented. With the script presented here, pixels in the final image will represent the most up-to-date data for the space they are located in.

Appendix C

ArcGIS Pro Models

This appendix contains the models built for Process 2.

---

fig. 14

Model 1

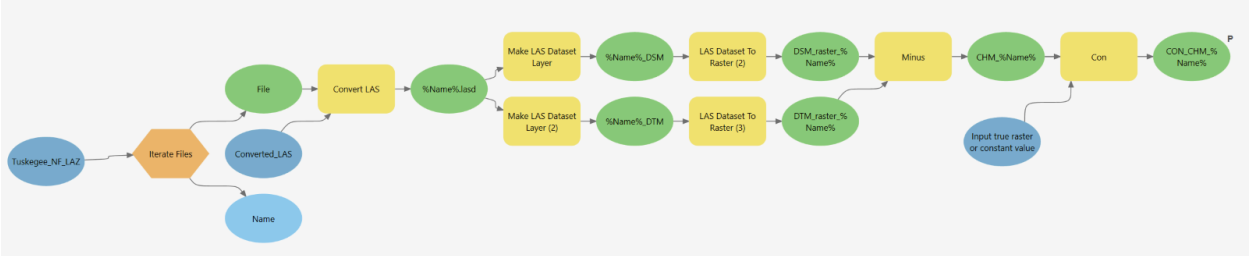

fig. 14 is the modelbuilder flowchart that performs the unpacking of LAZ files to LAS files, before interpolating DSM and DTM files and creating CHMs.

---

fig. 15

Model 2

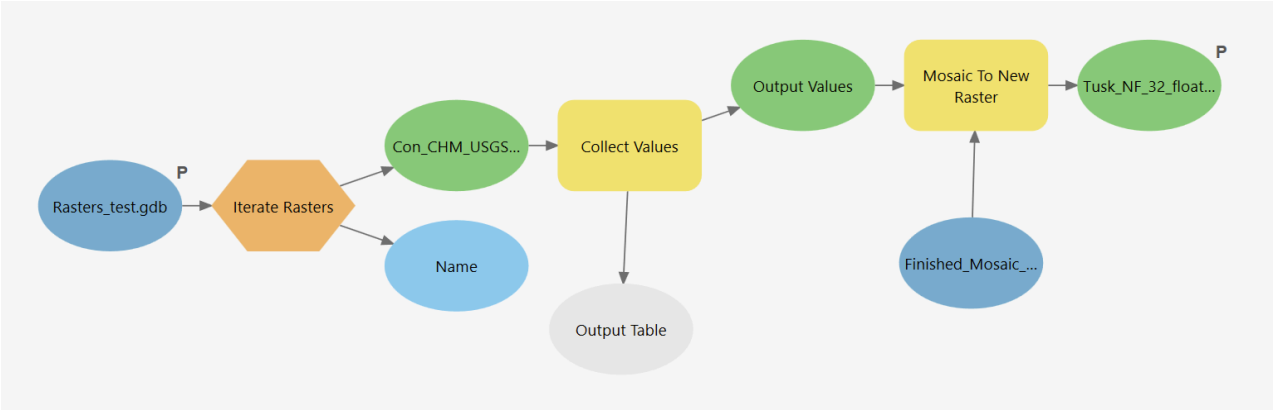

fig.15 Takes the completed CHMs and mosaics them into a unified CHM Raster.

---

fig. 16

Model 3

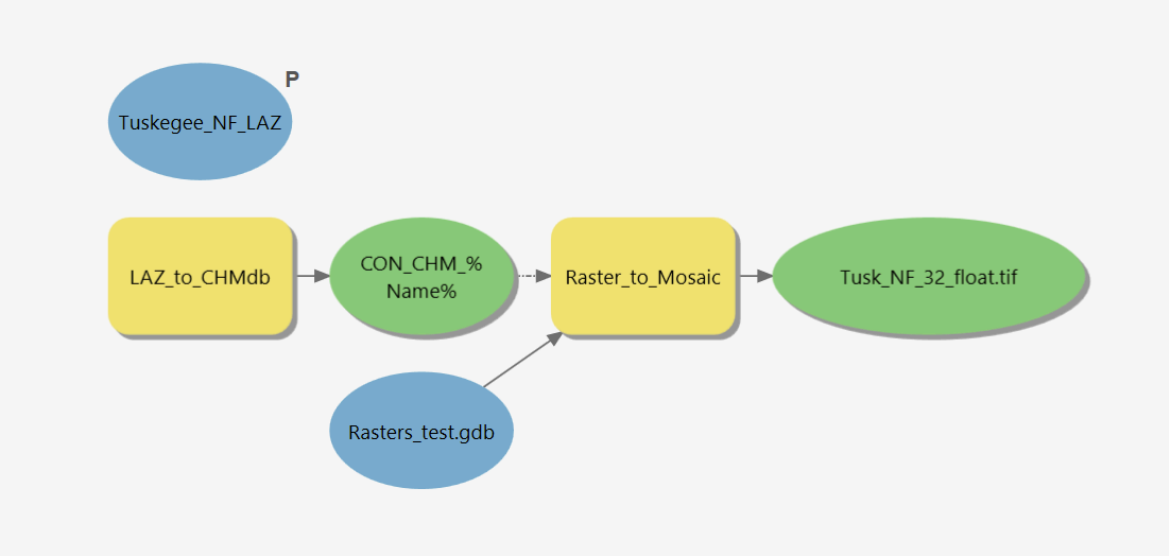

fig.16 is the Model builder parent proces that orchestrates models 1 and 2.

Appendix D
Statistical Analyses Python Scripts

fig. 17

Example Paired T-Test Script

```
import rasterio
from scipy.stats import ttest_rel

# File paths to the raster datasets produced by Process A and Process
B
process_a_file = 'FUSION_CHM.tif'
process_b_file = 'Pat_CHM.tif'

# Open the raster datasets using rasterio
with rasterio.open(process_a_file) as process_a_src,
rasterio.open(process_b_file) as process_b_src:
    # Read raster data as numpy arrays
    process_a_data = process_a_src.read(1, masked=True)  # Assuming
single-band raster
    process_b_data = process_b_src.read(1, masked=True)

    # Perform the paired samples t-test
    t_statistic, p_value = ttest_rel(process_a_data.flatten(),
process_b_data.flatten())

# Output the results
print("Paired Samples t-test Results:")
print("T-statistic:", t_statistic)
print("P-value:", p_value)

# Interpret the results
alpha = 0.05  # Significance level
if p_value < alpha:
    print("Reject null hypothesis: There is a significant difference
between Process A and Process B.")
```

```
else:
    print("Fail to reject null hypothesis: There is no significant
difference between Process A and Process B.")
```

fig. 17 shows an example python script used to perform a paired T-test between a CHM raster created by an experimental process (Process 3 in this case) and a CHM raster created by the Benchmark process.

fig. 18

Example Wilcox Signed-Rank Test Script

```
import rasterio
from scipy.stats import wilcoxon

# File paths to the raster datasets produced by Process A and Process
B
process_a_file = 'FUSION_CHM.tif'
process_b_file = 'Pat_CHM.tif'

# Open the raster datasets using rasterio
with rasterio.open(process_a_file) as process_a_src,
rasterio.open(process_b_file) as process_b_src:
    # Read raster data as numpy arrays
    process_a_data = process_a_src.read(1, masked=True)  # Assuming
single-band raster
    process_b_data = process_b_src.read(1, masked=True)

    # Flatten the arrays for the Wilcoxon signed-rank test
    process_a_flat = process_a_data.flatten()
    process_b_flat = process_b_data.flatten()

    # Perform the Wilcoxon signed-rank test
    _, p_value = wilcoxon(process_a_flat, process_b_flat)

# Output the results
print("Wilcoxon Signed-Rank Test Results:")
print("P-value:", p_value)
```

```
# Interpret the results
alpha = 0.05  # Significance level
if p_value < alpha:
    print("Reject null hypothesis: There is a significant difference between Process A and Process B.")
else:
    print("Fail to reject null hypothesis: There is no significant difference between Process A and Process B.")
```

fig. 18 shows an example python script used to perform a Wilcox Signed-Rank Test between a CHM raster created by an experimental process (Process 3 in this case) and a CHM raster created by the Benchmark process.

Glossary

Acronym	Definition
3DEP	3-D Elevation Program
AOI	Area of Interest
ASCII	American Standard Code for Information Interchange
ASPRS	The American Society for Photogrammetry & Remote Sensing
CCM	Canopy Cover Models
CHM	Canopy Height Models
DSM	Digital Surface Model
DTM	Digital Terrain Model
FCM	Forest Canopy Models
FOSS	Free and Open Sorce Software
FUSION	A suite of software developed by station scientists to visualize and analyze airborne LiDAR data
GEDI	Global Ecosystem Dynamics
GeoTIFF	A public domain metadata standard which allows georeferencing information to be embedded within a TIFF file
GIS	Global Information System
ISS	International Space Station
JSON	JavaScript Object Notation
LAS	LIDAR Aerial Survey
LAZ	LASzip
LiDAR	Light detection and ranging
NumPy	Numerical programming package for the Python programming language
PDAL	Point Data Abstraction Library
QA	Quality Assurance
QGIS	Open source desktop geographic information system software

Acronym	Definition
TIFF	Tagged image file format
USFS	United States Foresty Service