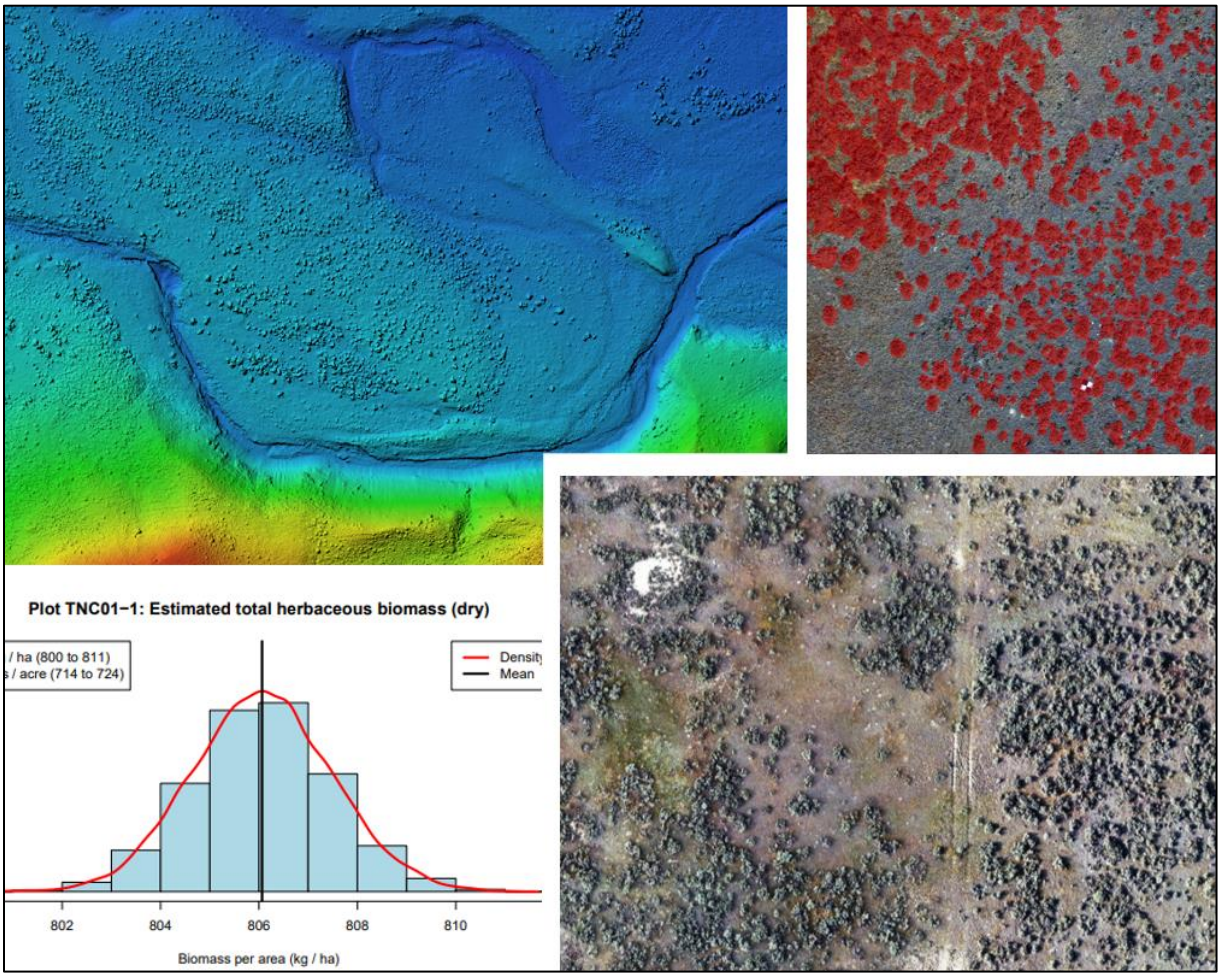
APPENDIX A

UAV DATA PROCESSING PIPELINE V0.1

Technical Guide

June 2019



Prepared for:
Brian Martin
Montana Grasslands
Conservation Director

THE NATURE CONSERVANCY
MONTANA

Contract P103095

Table of Contents

Figures:

1.0 Introduction

The Unmanned Aerial Vehicle (UAV) data processing pipeline (hereafter Pipeline) is an automated system to generate spatial data from UAV imagery. The Pipeline was developed to support rangeland monitoring efforts for the Phillips and Valley Counties Candidate Conservation Agreements with Assurances program (CCAA).

The goal of this project was to develop a platform that can efficiently extract ecologically relevant data from UAV photography with minimal analyst input and processing. The data products from this first version of the Pipeline are limited but can be improved or expanded in time with further development.

This technical document outlines the use and implementation of the Pipeline. The development of the science products and discussion of modeling methodology are discussed in an accompanying science report.

The Pipeline produces the following outputs in spatial and tabular forms:

1. Sagebrush cover (%)
2. Sagebrush plant centroid point and sagebrush density (plants per acre)
3. Sparse/bare ground cover (%)
4. Herbaceous biomass
5. Grass height
6. Canopy height model
7. Bare earth elevation model (DEM)
8. Digital surface model (DSM)
9. Orthoimagery

The data and models generated by this system are aggregated in two ways. First, the spatial data is delivered with an ArcGIS Map Document (*.mxd* file) to allow for easy interrogation by TNC staff in ArcGIS desktop. Second, plot-level statistics are provided in a series of figures and spreadsheets in pdf and CSV formats.

High-level operation of the Pipeline is controlled through R and subroutines are written in multiple languages, including python and shell scripting. Currently, the Pipeline has been designed for implementation in Windows 10 due to dependence on ArcGIS geoprocessing tools and LasTools. In the future, the system could be ported to Unix through integration with ArcGIS Server and using LasTools through Wine or other emulation packages.

The dependencies for the Pipeline are extensive, as currently there is not a single solution to implement photogrammetry and data modeling. The dependencies for this pipeline (excluding R Libraries) include:

1. PhotoScan Photogrammetry software (also called Metashape)
2. ArcGIS geoprocessing tools (spatial analyst extension)
3. Python 3 and Anaconda
4. LasTools
5. TensorFlow and Kera's
6. CUDA/CUCNN libraries for deep learning frameworks on GPU
7. GDAL Libraries

Implementation of the Pipeline has been tested on a Windows 10 machine with 12 threads, 64GB of ram, and an Nvidia 980Ti GPU. Much of the processing is multithreaded and the photogrammetry software and deep learning frameworks require a CUDA compatible GPU.

The Pipeline is a data science tool and not an enterprise solution. The development of this Pipeline was the outgrowth of research into using UAVs as a monitoring tool and is provided to TNC Montana to complement ongoing research on rangeland monitoring practices. The system has been tested on roughly 120 UAV plots in Phillips and Valley Counties, but no warranty is provided regarding the future maintenance of the Pipeline. The accuracy, quality, and bias of the data products are discussed in an accompanying science report and apply only to locations to Phillips and Valley Counties, Montana.

This remainder of this document summarizes the Pipeline to allow for implementation by TNC Montana staff. It includes user instructions for generating input, a discussion of the general workflow/ implementation, and more detailed instructions to help IT staff implement the system either locally or on a cloud-based virtual server.

2.0 Term of Use

This product has been developed for the TNC Montana Grasslands Program to support the CCAA program in Phillips and Valley Counties. The products of this system, including all code and system outputs, can be used and modified by TNC as needed. This product, associated code and models cannot be shared, duplicated, replicated, or transferred to any other entity without the written consent of Terra Analytics.

3.0 UAV Data Acquisition

The Pipeline and modeling tools presented here require uniform imagery collection to produce good data products. A wider discussion and rationale for these requirements are provided in the accompanying science report. The pipeline should be flexible enough to handle some deviation from these parameters but may fail or produce unexpected results if inputs stray too far from the intended flight and imagery acquisition parameters. These rules include:

- UAV Ground Sampling Distance (GSD) should be approximately 2cm.
- Image side-lap and overlap should be greater than 75% between images.
- UAV Camera orientation should be nadir.
- Image collection area (plot) should be roughly rectangular or square.
- Plot sizes should be roughly 25 acres. Larger plots may work but could fail due to resource requirements during processing.
- Photo acquisition must avoid image overexposure.
- Photos must have metadata (EXIF) with camera parameters and GPS location.
- Jpeg format is preferred. Other formats (e.g. RAW) will need to be converted before processing.

The data processing pipeline was tested on more than 100 25-acre UAV plots in Phillips and Valley Counties Montana. We used a DJI Phantom 4 Pro to collect imagery. Automated flight plans were created with the Drone Deploy App. A UAV platform with a similar or better-quality camera sensor should produce usable data.

Finally, the quality of the data products will reflect the quality of the input imagery. Lighting conditions, camera ISO, and image blur all impact the quality of the final models. UAV operators should familiarize themselves with image acquisition best practices before collecting data.

4.0 User Inputs and Program Execution

User inputs to the system are comprised of project folders containing geotagged UAV images. The location of the master input directory should be specified to the user by the administrator of the Pipeline (see implementation section). Processing is initiated by executing the master script found in the project */src* directory using one of two methods:

> R CMD BATCH *%Projectfolder%*/src/Master.R

or

> Rscript *%Projectfolder%*/src/Master.R

Where *%Projectfolder%* is the path to the project folder. The script can either be executed manually after placing the project folders into the input directory, or a task scheduler can be used to execute the script when new project folders are found. These choices are at the discretion of the administrator.

4.1 User inputs

Input to the Pipeline consists of one or more project folder(s) containing UAV images (**Figure 1**). After collecting data, the UAV operator should put all images from a single plot into a folder named for the project location (e.g. 'BLM43'). Project folders should not contain subfolders.

Use either Jpeg or Tiff image formats as inputs to the Pipeline. If you capture data in Raw, you will need to convert the images first. If you pre-process the imagery, make sure that the geographical and camera metadata is passed to the EXIF tag in the final image. The EXIF data is typically viewable through the image-file properties (**Figure 1**), or with common digital photo management software.

Users should visually check all images before passing the data to the Pipeline. Remove images that are blurry or that do not represent crisp nadir images of the plot. Many UAVs will collect an image at power-on and if these extra images are included in the Project folder they may crash the photogrammetry software or produce expected results.

The Pipeline will search for new project folders to processing during startup but will not find new project folders added after the Pipeline as started. The best practice is to place all project folders for processing into the master input directory prior to launching the Pipeline.

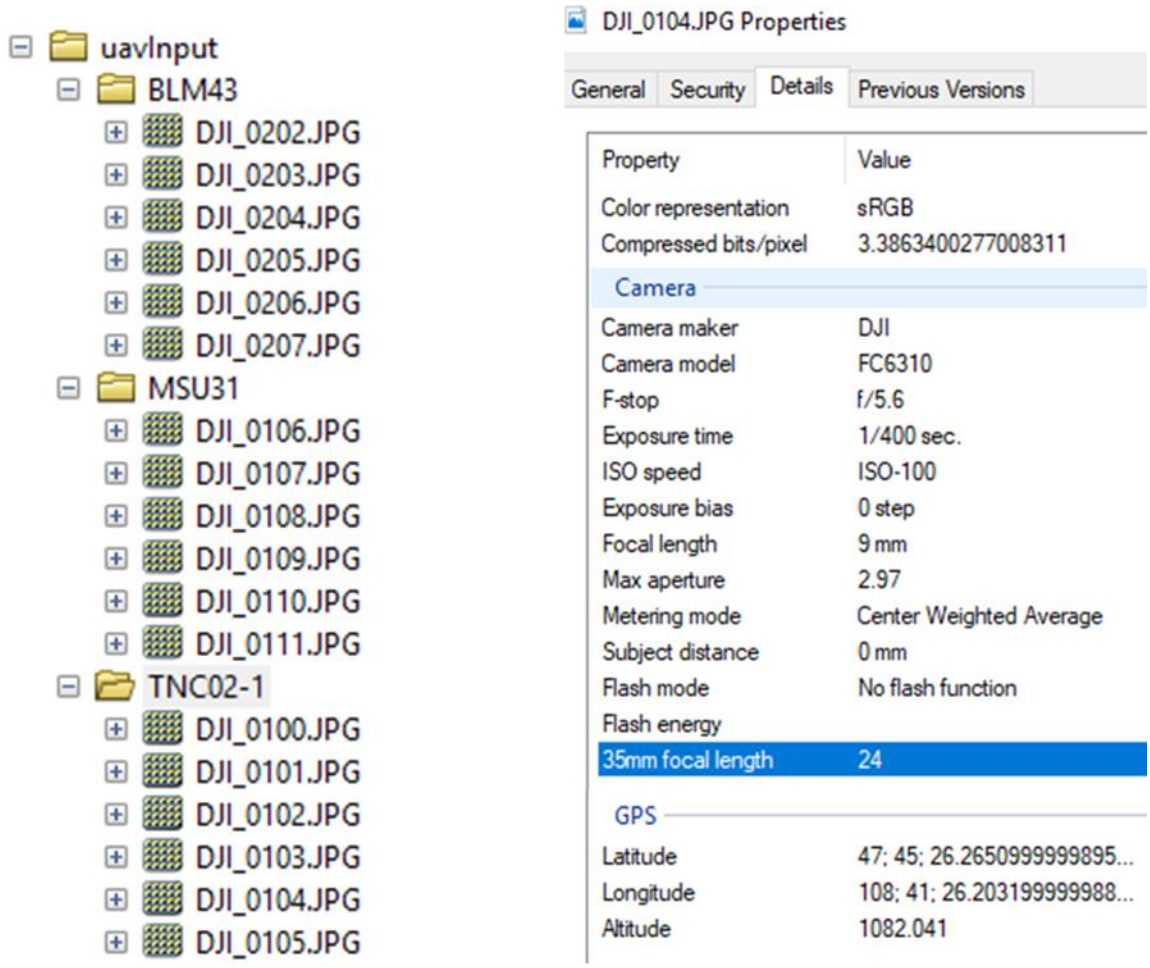


Figure 1: Input directory structure and image properties

5.0 Program Outputs

Following successful completion of Pipeline processing, a project folder will be placed in the master output directory. The location of the master output directory is specified by the administrator (see implementation section). Each output project folder will correlate with the input folder of the same name. The basic structure and composition of the output folders are shown in **Figure 2**.

The size of an output project folder will be ~ 10 GB for standard output or ~ 25 GB when photogrammetry data is included. These size estimates are based on a plot size of roughly 25 acres with a GSD of 2 cm.

Each project folder should contain four subdirectories:

logs: Processing and logs.

products*:* Geospatial data products from the Pipeline

rawImagery: Input imagery.

reports: Data summary reports in pdf and CSV formats.

photogrammetry: Raw photogrammetry project files and output (optional export).

The ***logs*** directory contains text logfiles generated by model subroutines that are not directly performed in R (e.g. python processing by PhotoScan and ArcPy, and LasTools functions). If the processing pipeline fails, these files should be useful for debugging. Errors that occur within R core functionality and functions should produce error messages that are printed to *standard out* or the *Rout* file, depending on how the script is implemented.

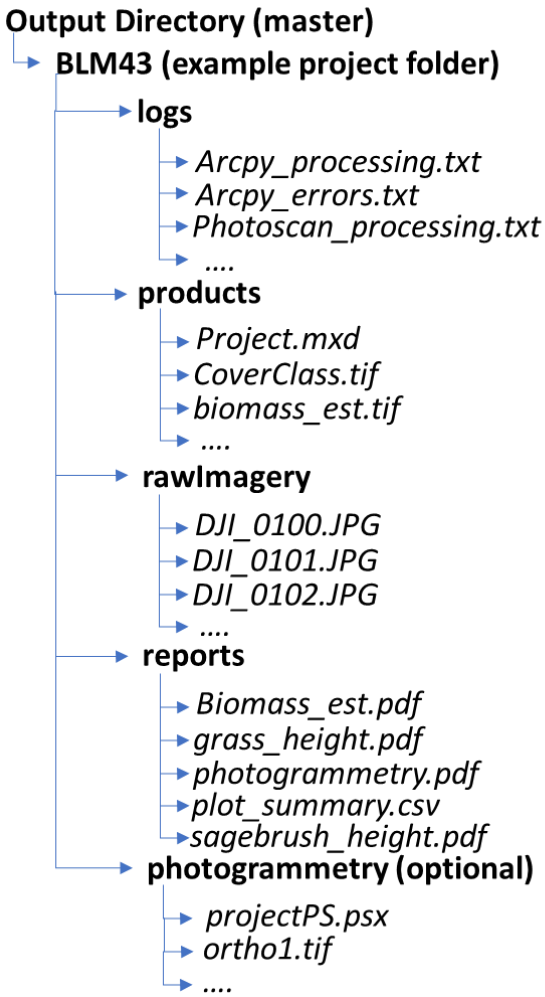


Figure 2: Output project folder structure

The ***products*** directory contains the primary geospatial products, including:

aoi.tif: Area of interest (plot boundary) in raster format.

aoiPoly.shp: Area of interest (plot boundary) in vector format.

biomass_est.tif: Herbaceous biomass in grams per meter squared.

Cameras.shp: Point vector file showing the image center point of input data.

chm.tif: Preliminary canopy height model uncorrected for structural anomalies.

***chm2.tif*:** Final canopy height model corrected by neural network bare ground algorithm.

chm2_hillshade.tif: Hill shade of chm2.tif useful for visualizing data in ArcGIS.

CoverClass.tif: Cover class output from the neural network algorithm, includes sagebrush cover and bare ground cover classification.

dem.tif: Digital elevation model from point cloud processing.

dem_hillshade.tif: Hill shade of dem.tif useful for visualizing data in ArcGIS.

dsm.tif: Digital surface model output from photogrammetry software.

grassht_est.tif: Grass height estimates in meters.

ortho1.tif: Orthoimagery using the dense point cloud as the elevation input.

ortho2.tif: Orthoimagery using the sparse point cloud mesh as elevation input. Better for visualization when plot features have angular shapes.

pointclound.laz: Filtered/cleaned dense point cloud in LAZ format.

Pointcloud_ground.laz: Point cloud used to generate DEM.

Project.mxd: An ArcGIS Map Project for visualizing data compatible with ArcGIS 10.3 and later. Users will need to click the 'full extent' icon/button in ArcGIS desktop upon loading.

sage_plant.shp: Point Vector features identifying individual sagebrush plants (experimental).

The **rawImagery** directory contains the input imagery data to the Pipeline. The input folder is deleted upon successful pipeline execution. Users should retain the raw input imagery locally until they successfully download and verify the output project folder.

The ***reports*** folder contains summary information from the plot, including:

biomass_est.pdf: Estimated herbaceous biomass, grass biomass, and forb biomass for the plot. The grass and forb biomass estimates are experimental and may be grossly incorrect.

Grass_height.pdf: Plot showing the distribution of grass height across the plot.

Photogrammetry.pdf: Photogrammetry report for the plot.

plot_summary.csv: Tabular data of plot summary statistics.

sagebrush_height.pdf: Figure showing the distribution of sagebrush height based upon the maximum height of each individual plant.

The ***photogrammetry*** folder is an optional output that includes the PhotoScan project file, the raw orthoimagery, digital surface model, and point cloud data. By default, these data are not provided in the output, but can be included using a flag in the master processing script (see implementation section). These data could be useful as they have a broader spatial extent than the spatial data provided in the ***products*** directory. The Photogrammetry products add approximately 15 GB of data to the project folder.

6.0 Workflow Implementation and Processing

This section provides a basic description of how the Pipeline operates. This documentation is not exhaustive; rather, it is geared towards helping the program administrator and users understand how the Pipeline processes data at a high level.

6.1 Project folder structure

The project folder is comprised of four primary folders: *bin*, *src, input*, *output*. The script files to execute the Pipeline are provided in the ***src*** directory and binary files are stored in the **bin** directory. The ***src*** and ***bin*** directories should not be removed or separated from the project folder. The *input* and *output* folders can be moved to different local or network drives; configuring these directory pointers is discussed in **Section 6.3**.

6.2 Pipeline Workflow

The Pipeline comprises of two high-level subroutines. Upon executing ***master.R*** the Pipeline first checks for required external dependencies, scripts, and model files. If all required dependencies are found, the pipeline then checks for project input folders. If one or more input folders are found, the pipeline then iterates through each input project folder sequentially. (**Figure 3 – Preprocessing**).

Next, the candidate input project folder is checked for compatible imagery. If found, the imagery is copied to an intermediate processing directory, and the data processing pipeline is initiated (**Figure 3 – Data Processing**).

Data processing routines consist of 9 core functions, broken into 16 steps shown in **Figure 3**. First, the imagery is processed by photogrammetry software (Step 1), followed by point cloud filtering and cleaning (step 2). Next, the pipeline generates canopy height (CHM) and bare-earth elevation models (steps 3-4). Imagery and the CHM are passed to deep learning classifiers to produce the sagebrush and bare ground cover models (steps 6-7). The bare ground classification product is then used to update the CHM (step 8), and the cover models are aggregated into a single raster layer (step 9). The system then builds derivative model products (e.g. biomass, grass height, etc.) and extracts plot-level statistics (steps 10 – 12). Next, the pipeline produces spatial and report products (step 13-14). Finally, the completed project data is scrubbed of intermediate data and placed in the output folder (steps 15 -16).

The Pipeline then iterates through remaining input project folders.

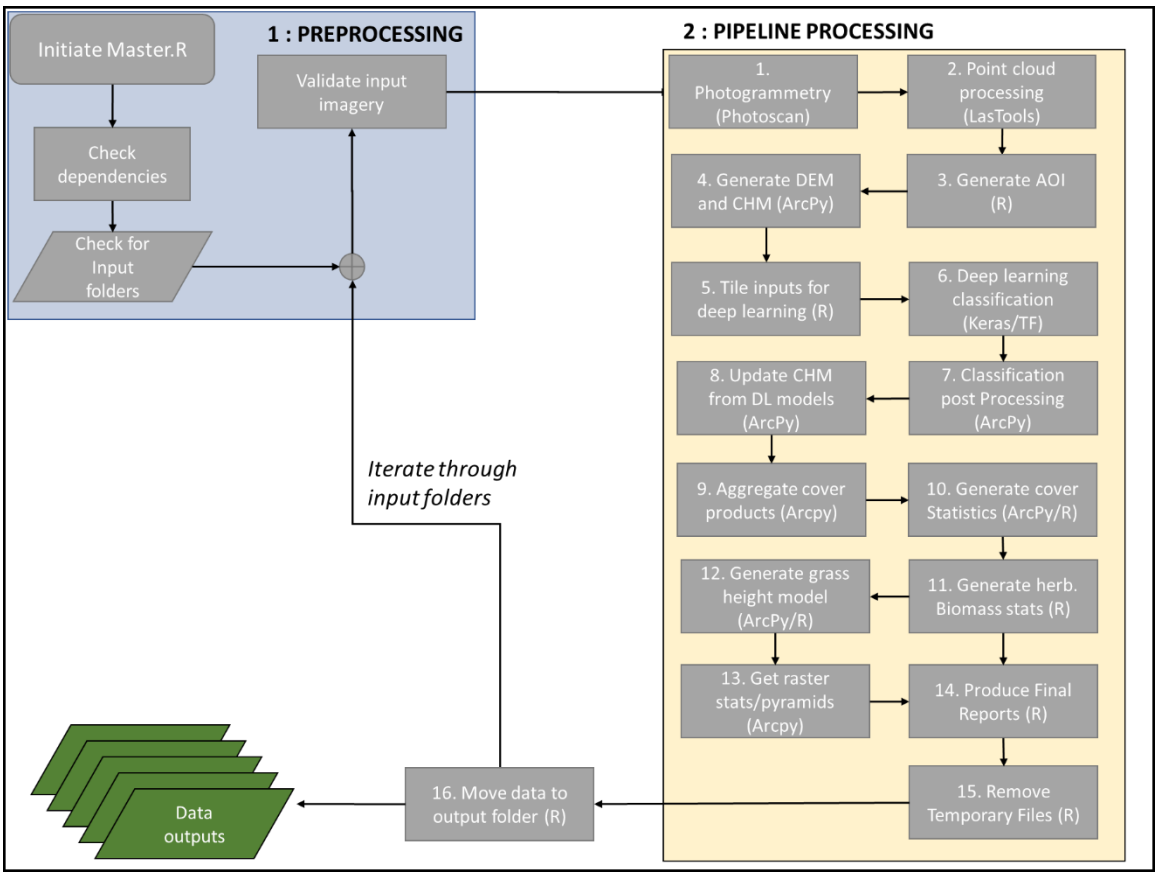


Figure 3: Pipeline workflow

Data processing routines are controlled by script files found in the ***src*** directory. Data processing routines performed directly by R libraries reside in the ***0_functions.R*** script. Functions and processing routines that are performed externally to R libraries (e.g. Python scripts) are included separately. The scripts are numbered to indicate the sequence in which they are executed but do not correspond directly to the steps shown in the workflow diagram (Figure 3).

6.3 Processing time and resource requirements

The time required to process an individual project folder is highly dependent on the number of CPU threads available for parallel processing and the type of GPU. We have tested the pipeline under two configurations to provide an initial estimate of the time required to successfully process a single project folder. The processing times provided below are based on a 25-acre plots with roughly 220 input images.

A processing time of approximately 8 hours is achieved with a 6 Core (12 Thread) Intel CPU – 4.00 GHz with 64 GB of ram and a Nvidia 980Ti CUDA compatible GPU. Photogrammetry and Point cloud filtering comprise most of the processing time, requiring 4.5 and 2.2 hours, respectively.

A processing time of approximately 4.5 hours was achieved with an AMD 16 Core (32 Thread) CPU – 4.00 GHz with 128 GB of RAM and a Nvidia 1080TI CUDA compatible GPU.

7.0 Implementation and Updating

This section includes information on installing and configuring the Pipeline. As mentioned above, the codebase has been tested on two Windows 10 systems using R-3.5.3.

7.1 Installation of R and supporting external libraries.

Prior to installing R library packages and configuring master variables, the following programs/libraries must be installed:

R (base): Available from CRAN (https://cran.r-project.org). Use version 3.5.3 to ensure compatibility.

RStudio: The RStudio IDE (https://www.rstudio.com/). Required for TensorFlow/Keras API.

Rtools: R tools for building packages. Available from CRAN. After install add binary location to the system path.

Python 3 and Anaconda: Download from Anaconda. Virtual Environment functionality is required to use Keras and TensorFlow in R. It is suggested that you add the anaconda python library directory to your system path.

PhotoScan Professional: Photogrammetry software from Agisoft. Version 1.4.5 has been tested successfully. The python module for newer versions (i.e. Metashape 1.5.x) may not be compatible with the current python processing script.

Nvidia CUDA Toolkit v9: Available from NVIDIA. Required for photogrammetry and Deep learning frameworks. See the RStudio TensorFlow installation page for additional information. Setup of the CUDA toolkit on cloud-based services (EC2/Azure) may require additional steps – see the platform documentation.

Nvidia cuDNN v7.0 driver: Available from Nvidia (requires login). These are neural network acceleration drivers used in the deep learning frameworks Setup of the cuDNN libraries on cloud-based services (EC2/Azure) may require additional steps – see the platform documentation.

GDAL: Geospatial Data Abstraction Libraries (https://gdal.org) can be downloaded directly as an installer or can be accessed through typical Unix package tools. For Windows 10 applications, the binary installers from *GISInternals* (http://www.gisinternals.com/) or the bundled GIS tools from OSGeo (https://trac.osgeo.org/osgeo4w/) generally work well.

LasTools: Available from Rapid Lasso. Used for Point Cloud Processing. Note that some OSGeo distributions will come with outdated versions LasTools binaries. Make sure your system path configuration does not use the OSGeo LasTools binaries by default.

ArcGIS Python API: ArcGIS geoprocessing tools are required. These can be accessed through a local install of ArcGIS Desktop or via Enterprise Server configuration. Check with your administrator. The python scripts used in this Pipeline were built and validated Using the Python API installed with ArcGIS Desktop 10.6.1.

7.2 Installing R library packages

To install required R packages, execute the ***Install_Required_Packges.R*** script found the ***src*** directory. This script will attempt to download compatible versions of all the required packages and run some additional installation scripts. This script should be run interactively and checked for errors. In particular. The *install_keras()* function is error prone and may require the administrator to view the online documentation to ensure the system is configured correctly for use with GPU and the *Anaconda* virtual environment. Solutions to common problems can also be found at their GitHub site.

Note that *hd5* model files for deep learning may not be compatible with newer versions of TensorFlow and the Keras API. If problems arise when attempting to load the model weights during step 6 of the data processing subroutine, ensure that you have Keras 2.2.0 and TensorFlow 1.9 libraries installed.

```
> sessionInfo()
R version 3.5.3 (2019-03-11)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows >= 8 x64 (build 9200)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] splines   parallel  stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] gam_1.16          abind_1.4-5       keras_2.2.0       doParallel_1.0.14 iterators_1.0.10
 [6] foreach_1.4.4     TileManager_0.3.0 shotGroups_0.7.4  rgdal_1.4-3       rgeos_0.4-2
[11] maptools_0.9-5    sp_1.3-1

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1          compiler_3.5.3      DEoptimR_1.0-8      base64enc_0.1-3     tools_3.5.3
 [6] boot_1.3-20         zeallot_0.1.0       jsonlite_1.6        lattice_0.20-38     Matrix_1.2-15
[11] yaml_2.2.0          mvtnorm_1.0-10      CompQuadForm_1.4.3  coin_1.3-0          libcoin_1.0-4
[16] mmap_0.6-17         stats4_3.5.3        grid_3.5.3          reticulate_1.11.1   robustbase_0.93-4
[21] R6_2.4.0            survival_2.43-3     foreign_0.8-71      multcomp_1.4-10     TH.data_1.0-10
[26] whisker_0.3-2       magrittr_1.5        tfruns_1.4          codetools_0.2-16    matrixStats_0.54.0
[31] modeltools_0.2-22   MASS_7.3-51.1       tensorflow_1.9      sandwich_2.5-1      KernSmooth_2.23-15
[36] zoo_1.8-5
```

Figure 4: R library packages and versions

7.3 Configuring the Pipeline

The code base was designed to be completely automated, with the total of user-supplied information passed to the program via input project folders. However, upon initial setup, it is necessary for the administrator to edit the ***Master.R*** script to provide the correct directory pointers and set processing parameters, including:

> ***tile.threads*** (line 11)*:* Takes numeric argument. Number of processing threads to use to tile data for deep learning. Typically, this should be the total number of available threads minus one or two.
>
> ***lastools.threads*** (line 12): Takes numeric argument. Number of processing threads used to filter/clean point cloud data. Experimentally, the best results were found when *lastools.threads*

equaled total system threads minus two. This variable is likely system dependent, as processing is very I/O heavy.

unet.threads (line 13): Takes numeric argument. Number of processing threads used to pre-process data for GPU deep learning models. Experimentally, increasing this number above 8 did not improve performance, but performance gain by tuning this parameter is likely system specific.

unet.batchsize (line 15): Takes numeric argument. Batch size of arrays fed to the deep learning model. The value of this parameter should be maximized based on the system hardware configuration. The maximum batch size number is a function of available GPU RAM. If using CPU to do deep learning predictions, the batch size can be much larger, but predictions will be much slower regardless.

remove.photogrammetry (line 17): Takes logical argument. Specify whether to remove the photogrammetry project data (raw rasters, raw point cloud data, and PhotoScan project) following processing. Setting to *TRUE* will save approximately 15GB of data storage per project.

programDir (line 21): Takes character string argument. File system location of the master project folder.

inputDirMaster (line 25): Takes character string argument. File system location for the master input directory.

outputDirMaster (line 26): Takes character string argument. File system location for master output directory.

scratchDir (line 27): Takes character string argument. File system location for data processing. Should be local to the maximum I/O bandwidth and minimize latency.

path.PhotoScan (line 31): Takes character string argument. File system location of the PhotoScan Pro installation.

path.lastools (line 32): Takes character string argument. File system location of the LasTools installation. *Note*: If OSGeo binaries are in your %PATH%, point cloud processing may fail due to executing older versions of LasTools binaries associated with the GDAL library installation.

path.arcgis.python (line 33)**:** Takes character string argument. File system location of the ArcGIS python executable.

7.4 Updating Model Files

The pipeline does not require updating after initial implementation. However, deep learning models will continue to be improved and updated by Terra Analytics. These files will be provided to TNC Montana as they become available. The model files, which contain the weights for the neural-network architecture, are distributed in the Hierarchical Data Format (HDF – extension *.hd5)* and can replace their corresponding copies in the *Data/* directory.

7.5 Licensing of programs/libraries used in the Pipeline

The Pipeline uses a combination of open source, closed source, and commercial software. PhotoScan, ArcGIS, and LasTools are commercial software. TNC already manages ArcGIS licenses via enterprise systems, so a discussion is not required. However, some comment is required in the case of PhotoScan and LasTools.

TNC Montana purchased a PhotoScan license for use with this project. The node-locked license can be moved among different systems but can only be activated on a single host at any given time. For PhotoScan 1.4.X, there is not the ability to activate/deactivate the license via the Python API. If TNC Montana GIS Analysts want to use PhotoScan license locally, they will either need to deactivate the license manually on the cloud server or come up with another solution with their IT administrators. Possible solutions include:

- Upgrading PhotoScan 1.4.x to Metashape 1.5.x and use the Metashape 1.5.x Python API to activate/deactivate the license. It is unknown how difficult it will be to migrate the current python code from 1.4.x to 1.5.x.
- Contact Agisoft regarding using a Floating License System.
- Provide the TNC Montana GIS analyst with instructions for managing the license manually on the cloud service.

LasTools is a combination of open source and commercial binaries. The LasTools terms of use specify that the program can be used at no-cost by TNC Montana for non-profit environmental work. During development, Terra Analytics spoke with Martin Isenburg (Rapid Lasso) to confirm that this project does not break LasTools terms of use, and a commercial license is not required. However, the free LasTools software used here does have limitations. To minimize the effects of these limitations, the point cloud processing pipeline is designed to be inefficient in how it tiles and buffers data. Moving to a commercial LasTools license and adjusting the tiling/buffering scheme would speed up point cloud filtering/cleaning by a factor of 2 - 4. Visit the Rapid Lasso website for pricing.

End document.