



---

MSU Graduate Theses

---

Spring 2019

## 3d Canopy Model Reconstruction from Unmanned Aerial System and Automated Single Tree Extraction


Hai Ha Duong

Missouri State University, Hai1979@live.missouristate.edu

As with any intellectual project, the content and views expressed in this thesis may be considered objectionable by some readers. However, this student-scholar's work has been judged to have academic value by the student's thesis committee members trained in the discipline. The content and views expressed in this thesis are those of the student-scholar and are not endorsed by Missouri State University, its Graduate College, or its employees.

---

Follow this and additional works at: <https://bearworks.missouristate.edu/theses>

 Part of the [Geographic Information Sciences Commons](#), [Programming Languages and Compilers Commons](#), and the [Spatial Science Commons](#)

### Recommended Citation

Duong, Hai Ha, "3d Canopy Model Reconstruction from Unmanned Aerial System and Automated Single Tree Extraction" (2019). *MSU Graduate Theses*. 3360.

<https://bearworks.missouristate.edu/theses/3360>

This article or document was made available through BearWorks, the institutional repository of Missouri State University. The work contained in it may be protected by copyright and require permission of the copyright holder for reuse or redistribution.

For more information, please contact [bearworks@missouristate.edu](mailto:bearworks@missouristate.edu).

**3D CANOPY MODEL RECONSTRUCTION FROM UNMANNED AERIAL SYSTEM**  
**AND**  
**AUTOMATED SINGLE TREE EXTRACTION**

A Master's Thesis

Presented to

The Graduate College of  
Missouri State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Natural and Applied Science, Geospatial Science and Computer Science

By

Hai Ha Duong

May 2019

# **3D CANOPY MODEL RECONSTRUCTION FROM UNMANNED AERIAL SYSTEM AND AUTOMATED SINGLE TREE EXTRACTION**

Geography, Geology and Planning

Missouri State University, May 2019

Master of Natural and Applied Science

Hai Ha Duong

## **ABSTRACT**

This project aims to develop and assess methodology for spatial modeling and extracting individual trees from high spatial resolution Digital Surface Model (DSMs) derived from unmanned aerial system (UAS) or drone-based aerial photos. Those results could be used for monitoring of vegetative response of forests, grasslands and vineyards to regional and localized fluctuations in climate and seasonality. The primary objective of this research is to extract 3D spatial information using drone-based aerial imagery through photogrammetric methods. UAS flights were taken place at phenologically critical times over several locations owned and managed by Missouri State University (MSU). The 3D DSM can be reconstructed from obtained images through photogrammetric software. And then a computer programming language (Python) is used to develop an algorithm to extract every single tree from 3D models. First, DSM data is imported into Python as a raster data layer. Next, the watershed segmentation algorithm with two different image filtering is used for single tree extraction, and the accuracy comparison is conducted between the two methods. The number of the trees can be used to calculate vegetative metrics and growth rate if multi-year data is available in the future. And time-series data can be correlated with climate fluctuation and seasonality. This study can provide a new approach for monitoring various types of natural and agricultural vegetation to aid in making short-term and long-term management decisions.

**KEYWORDS:** image segmentation, watershed, tree extraction, reconstruct 3D model, unmanned aerial system, UAS, DSM

**3D CANOPY MODEL RECONSTRUCTION FROM UNMANNED AERIAL SYSTEM  
AND  
AUTOMATED SINGLE TREE EXTRACTION**

By

Hai Ha Duong

A Master's Thesis  
Submitted to the Graduate College  
Of Missouri State University  
In Partial Fulfillment of the Requirements  
Master of Natural and Applied Science

May 2019

Approved:

Xin Miao, PhD. Prof., Thesis Committee Chair

Jun Luo, PhD. Prof., Committee Member

Lloyd A. Smith, PhD. Prof., Committee Member

Julie Masterson, PhD., Dean of the Graduate College

In the interest of academic freedom and the principle of free speech, approval of this thesis indicates the format is acceptable and meets the academic criteria for the discipline as determined by the faculty that constitute the thesis committee. The content and views expressed in this thesis are those of the student-scholar and are not endorsed by Missouri State University, its Graduate College, or its employees.

## ACKNOWLEDGEMENTS

I would like to express sincere gratitude to Prof. Dr. Xin Miao, Geography, Geology and Planning Department at the Missouri State University for providing me an opportunity to work on this topic along with the dedicate guidance and great help in so many difficult situations so that I can complete this study.

I would like to thank Prof. Dr. Lloyd A. Smith, who has been very helpful and has instructed me in numerous ways before and during my thesis. Special thanks and appreciation to Prof. Dr. Jun Luo. His valuable support as one of my thesis committee was vital for successful completion of my master thesis. I am very grateful to the members of Geography, Geology and Planning Department, especially Prof. Dr. Douglas R. Gouzie, for their kind co-operation and support which also helped me in completion of my thesis.

I would also like to thank my family and friends for all their advice and encouragement during the thesis.

## TABLE OF CONTENTS

Introduction	Page 1
Study site	Page 4
Equipment	Page 5
Programming language	Page 5
Methodology	Page 7
3D modeling	Page 7
Image segmentation and classification	Page 16
Results and discussion	Page 28
Conclusion	Page 33
References	Page 35
Appendices	Page 38
Appendix A. Processing parameters	Page 38
Appendix B. Source code with mean-shift filtering	Page 40
Appendix C. Source code with Gaussian filtering	Page 42
Appendix D. Source code of trees height calculation	Page 45

## LIST OF TABLES

Table 1. Control point RMSE	Page 11
Table 2. Segmentation results	Page 29
Table 3. RMSE of selected points	Page 32

## LIST OF FIGURES

Figure 1. The conceptual model of original research project	Page 3
Figure 2. Study site at Journagan Ranch	Page 4
Figure 3. General flowchart of 3D canopy surface generation and tree extraction.	Page 7
Figure 4. Photos alignment in Agisoft PhotoScan	Page 8
Figure 5. Camera information, locations and image overlap	Page 9
Figure 6. Image residual distribution.	Page 10
Figure 7. GCPs location	Page 11
Figure 8. 3D dense point cloud of the study site	Page 12
Figure 9. 3D Polygonal mesh of the study site	Page 12
Figure 10. Texture or orthophoto of the study site.	Page 13
Figure 11. DSM generated from dense cloud	Page 14
Figure 12. The exported DSM to geotiff format	Page 15
Figure 13. 3D visualization of DSM	Page 15
Figure 14. Canopy height model of the study site	Page 19
Figure 15. Result of thresholding	Page 22
Figure 16. Result of segmentation when using mean shift filtering	Page 24
Figure 17. Result of local maxima classification after applying Gaussian filter	Page 26
Figure 18. Image segmentation result	Page 26
Figure 19. 3D scatter plot of local maxima and trees height	Page 27
Figure 20. Results from two pre-processing filters	Page 29
Figure 21. Result when applying mean-shift filters	Page 29
Figure 22. Difficulty of segmentation	Page 30
Figure 23. Sum of Square Error graph	Page 31



## INTRODUCTION

The application of modern technology in environment monitoring is growing rapidly in the past ten years. Geospatial science is one of the areas where many modern technologies are applied and has grown stronger together with the development of science and technology. Specifically, remote sensing of vegetation becomes an important technique to monitor the vegetation growth rate that largely affected by climate change over seasons and years. It is important to monitor the vegetation change process so that timely response is taken before it is too late. Previously, scientists have used remote sensing satellites or airplanes to take images for tracking the variability of the vegetation. However, the cost of this method of implementation is high by purchasing satellite images, aerial photos, or getting digital data by LiDAR, the frequency of updating data is low, and the resolution of images is not high enough and affected largely by weather conditions. (Bertram *et al.*, 2014). This surveying process, therefore, lacking consistency. Another method is the analysis of satellite images has been using with higher resolution. For the purpose of observation, these images have to have accuracy, frequently updated and there is no cloud cover in the relevant spectrum bands. The images with above mentioned technical requirements are very expensive and therefore also very difficult to use for processing and interpretation. To observe the regular overall changes on the large area, remote sensing method using satellite images is still being considered as the optimal choice.

In the past decade, 3D modelling of landscape based on photogrammetry becomes popular with the rapid development of computing power. Among these researches, most aerial photos were captured by Unmanned Aerial Vehicles (UAVs) or drone-based systems.

The use of UAV has been broadly used in the field of geoscience for monitoring changes in terrain, vegetation, and urban landscape. The image data obtained in this way usually has high resolution. The increased intelligence of the flying device makes the process easier and does not need to be manually operated for a long time. In addition, the weather conditions do not heavily affect the operation of the unmanned aircraft system even in a cloudy day or sunny day. Especially, the use of unmanned aircraft to take aerial photos is convenient for a small study site with a much lower cost than LiDAR system. Moreover, technology is being used in unmanned aerial vehicles can support the accessibility of the area or viewpoint which previously were inaccessible, especially in the areas are often considered dangerous for local scientists such as embankment, steep cliff, easy subsidence terrain and the areas with high density of vegetation. Besides that, at the same time, it can give users the photographs and video images or also can have such a substantial support for media operations and archives.

Previously, there were many researches on image processing including vegetation extraction, canopy cover detection, and species recognition. However, those studies have mainly used 2D images for classification. This research project will use the drone-based aerial imagery through photogrammetric methods, reconstruct the 3D digital surface model (DSM), which can be used for visualization, terrain parameter extraction, and relief map generation. It helps the terrain analysis in geomorphology and physical geography, and the 3D model can be easily exported to a 3D printer. Specifically, the DSM can be used for vegetation analysis in this study.

This study is a part of a more comprehensive research of modeling the response of a black walnut -dominant mixed hardwood stand to seasonal fluctuations in climate. In the original proposed research project, we use UAS flights to gather hyperspectral and RGB aerial data during critical phenological shifts in the growing season and try to correlate vegetation stress

metrics and select seasonal climate variables (Figure 1). One of the challenges is to extract the trees from canopy surface (DSM) with a high accuracy instead of manual counting. It can significantly reduce the image processing time and increase the reliability.

In the current study, I will mainly generate a high-resolution 3D canopy model through aerial photogrammetry, and further develop an algorithm to count the individual trees with a reasonable accuracy from DSM.

The study includes the following objectives:

1. After acquiring aerial photos from a drone system, 3D DSM is generated through photogrammetric software (Agisoft PhotoScan Professional ver.1.2.6.).
2. Develop a watershed segmentation algorithm for automated single tree extraction from DSM through Python programming.
3. Assess and compare the accuracies of results from two pre-processing filters.

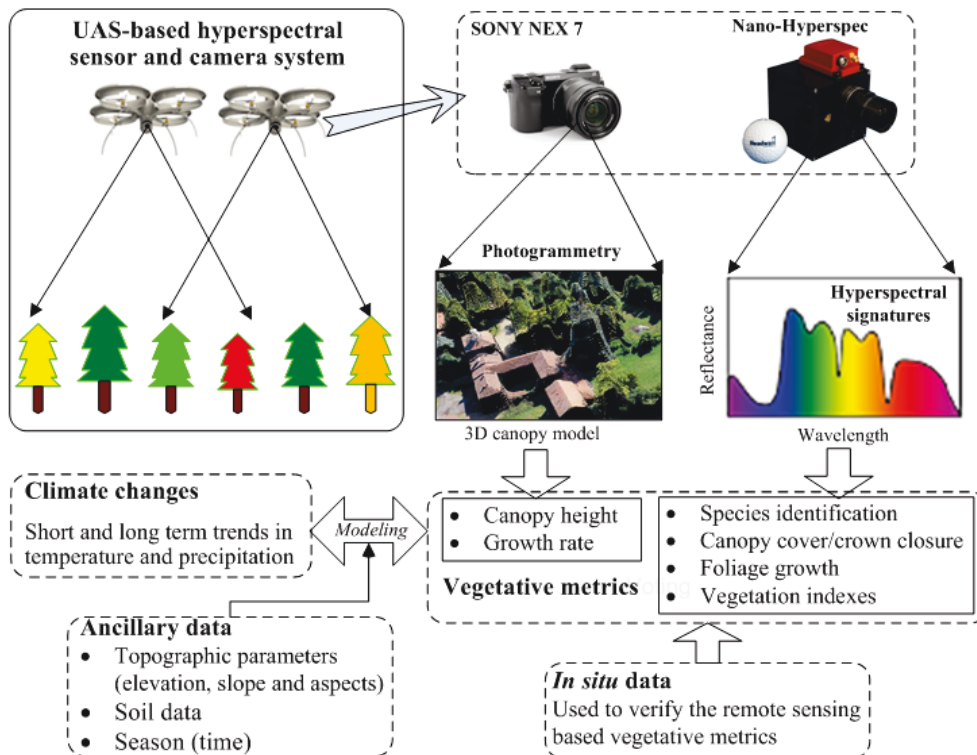


Figure 1. The conceptual model of original research project.

This study can provide a new approach for monitoring the stress of various natural and agricultural vegetation so as to aid in making short-term and long-term management decisions. It can further help developing a statistical model to correlate tree growth rate with climate changes in Missouri.

### Study Site

The study site is in Journagan Ranch, including several grassland, forestland, and vineyard locations owned and managed by Missouri State University (MSU). The chosen study site located near Mountain Grove, MO (Figure 2). A high abundance of black walnut (*Juglans nigra*) is dominated in this site. Black walnut is an appropriate environmental indicator of climate-induced stress due to its sensitivity to drought, which is a property from its evolutionary adaptation to semi-moist soils. Most importantly, the black walnut trees were planted approximately 60 years ago without significant human interference. The study site is relatively flat, and most of trees form clusters and develop a dense canopy. Since it is quite time consuming and labor intensive to count the trees in the field, a computer-aided assessing approach is needed to acquire a relatively reliable and accurate result.

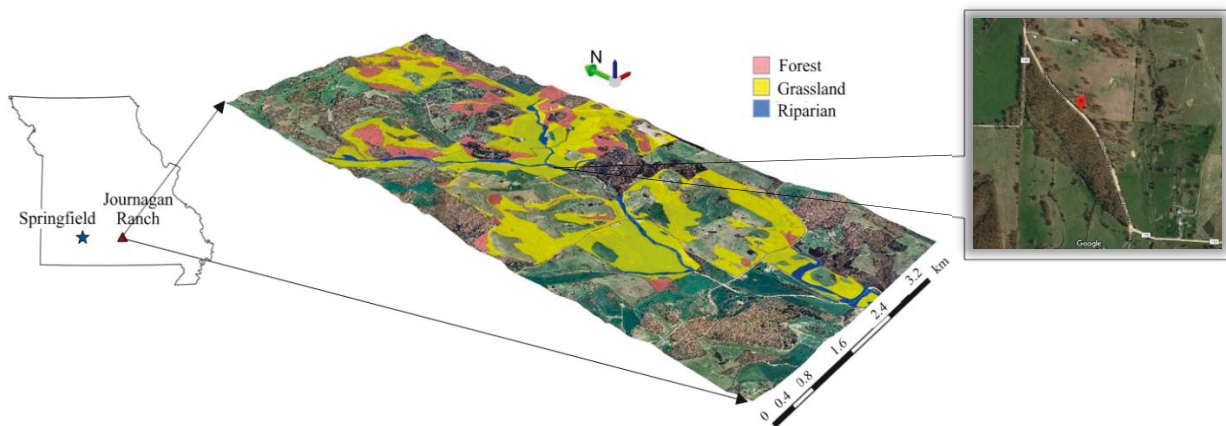


Figure 2. Study site at Journagan Ranch (coordinates: 37°02'17.0"N 92°15'11.7"W)

## **Equipment**

A Sony a7R II camera was mounted to a DJI Spreading Wings S1000+ airframe using a Pixhawk autopilot board, and the flight was conducted on June 6, 2017. The attitude of the UAV was measured by the XSens MTi-G-710 onboard IMU, which recorded the GPS location, roll, pitch, and yaw of the aircraft during the flight. Flight plans were designed with over 80% overlap between neighboring photos on the same flight strip. Ground data including diameter at breast height (DBH), stand-level statistics were collected in the previous works. And pre-defined Ground Control Points (GCPs) were collected using a handheld Trimble Nomad unit with a Trimble Pro 6H receiver for aerial photo registration and accuracy assessment. The geospatial accuracy is sub-meter under WGS 1984 (UTM Zone 15N) projection system (Kim *et al.*, 2016; Gupta and Shukla, 2017). GCP marks on the ground can be visually checked on aerial photos. The flying height is 120m with a constant speed of 10m/s. The UAS platform has retractable landing gear, vibration dampers, angled arms, and a gimbal mount that offers a 360-degree view from camera.

## **Programming Language**

There are many programming languages are being used which have different characteristics, such as imperative, modular, object-oriented, functional, relational, constrain and logical (Sebesta, 2002). Among them, Python is an open source language with a good object-oriented support and a rich library set (Caeiro *et al.*, 2014). The most popular and powerful libraries which have been using in geoscience are Geospatial Data Abstraction Library (GDAL), OpenCV, Scikit-image, SciPy and RasterIO. Moreover, the Python programming language is supported in ArcGIS - the actual industrial standard in geospatial science field. Besides, the

integrated numeric processing interface module has been developed which provides capabilities similar to Matlab and other array-based languages (Caeiro *et al.*, 2014). Therefore, Python programming is used for this study.

## METHODOLOGY

The process of 3D model reconstruction begins with flight planning, aerial photo collection, and then followed by dense point cloud generation through photogrammetric software, polygonal mesh generate, and Digital Surface Model (DSM) generation. Next, Python is used to conduct image segmentation and classification, and each tree can be finally extracted. The general flowchart of 3D canopy surface generation and tree extraction is illustrated in Figure 3.

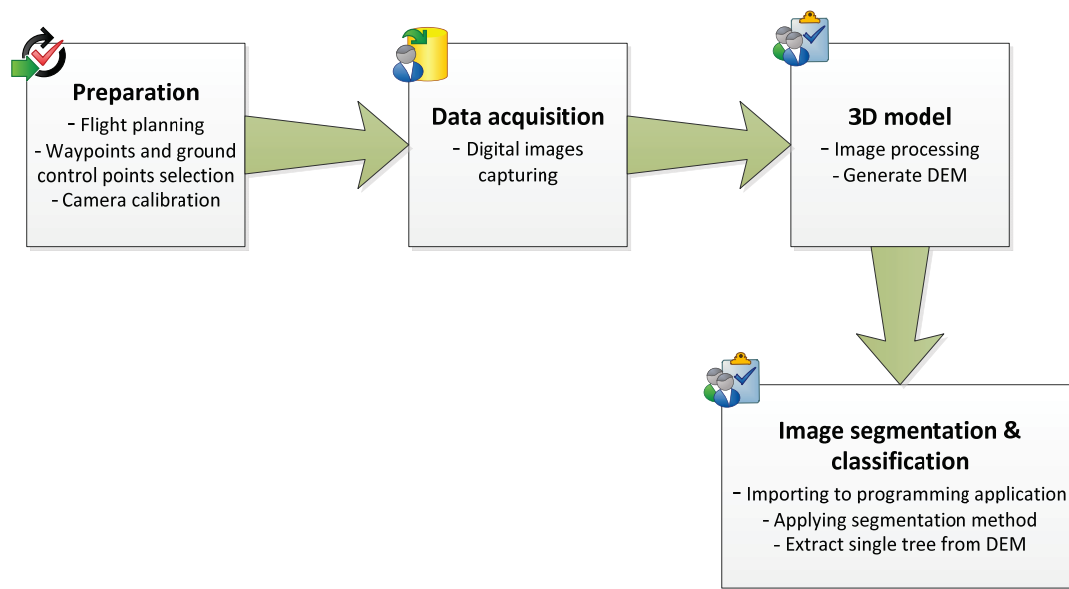


Figure 3. General flowchart of 3D canopy surface generation and tree extraction.

### 3D Modeling

The 3D geometry regeneration including the following main steps (Gupta and Shukla, 2018)

1. Photogrammetry or Structure-from-Motion (SfM): the attitude of camera such as position, orientation, and focal length should be reconstructed by tie points between images. Then a sparse point-based 3D presentation can be generated.

2. Multi View Stereo (MVS): uses the estimated camera parameters to find the visual relationships between images, and a dense 3D geometry can be reconstructed.
3. Surface Reconstruction: surface mesh has been produced by taking dense point cloud as input data.

**Image Processing.** All the obtained aerial images have been used as input data for the photogrammetry software (Agisoft PhotoScan 1.2.6). GPS information has been tagged and recorded in each image file with the Exchangeable image file format (Exif). Starting from these initial positions, the images alignment are conducted through matching tie points. Figure 4 shows an alignment for all taken images in the image processing software.

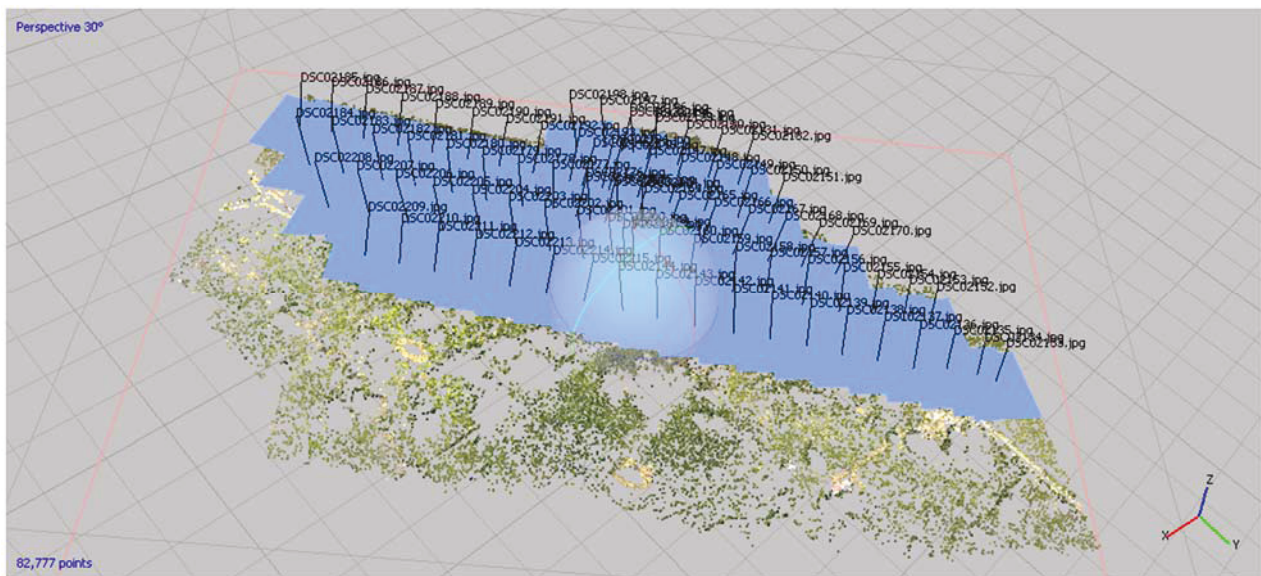


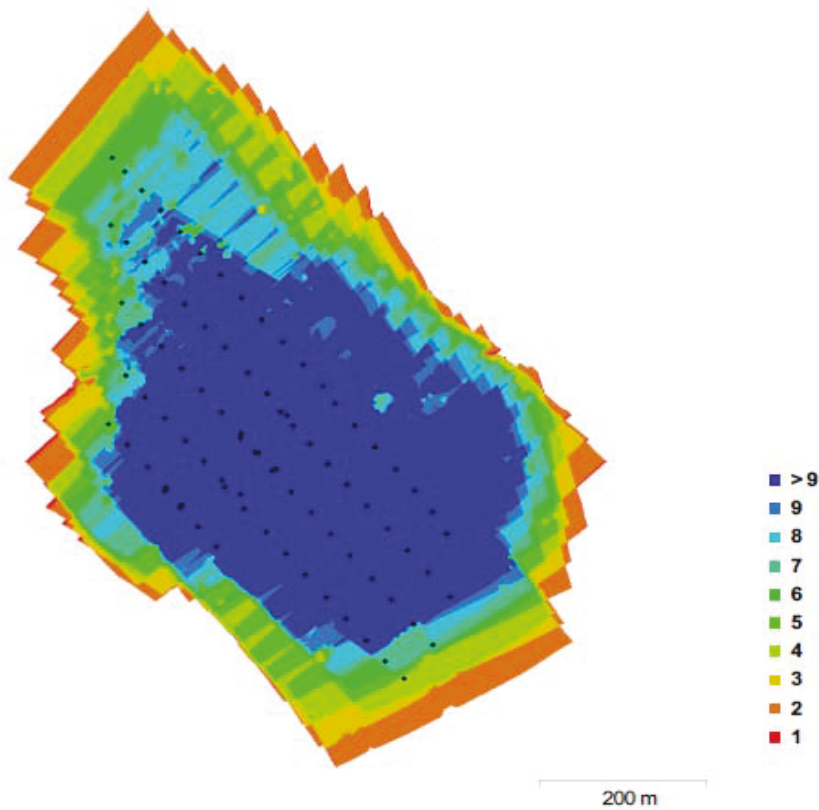
Figure 4. Photos alignment in Agisoft PhotoScan.

One crucial step in the 3D reconstruction procedures is checking the overlapping of obtained images to ensure that there are at least two images for any given ground target. Figure 5 below shows the lists of focal length, resolution, image size, altitude of the camera, magnification of images and the size of each pixel after image alignment together with camera



locations and overlapping of the images. Figure 6 shown the image registration residual or alignment errors of the aerial photos.

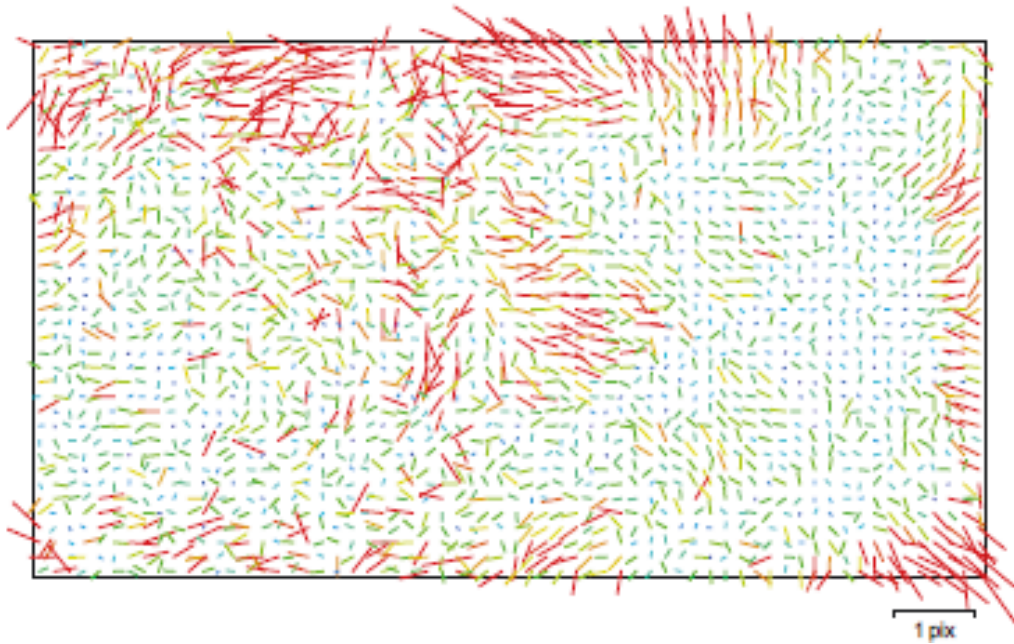
The coordinate of GCPs are used for 3D model geo-registration and accuracy assessment. The GCPs in the study site is shown in Figure 7. The Root-Mean-Square Error (RMSE) of GCPs are calculated and illustrated in Table 1.



Number of images:	85	Camera stations:	85
Flying altitude:	155 m	Tie points:	82,777
Ground resolution:	3.77 cm/pix	Projections:	234,411
Coverage area:	0.4 km <sup>2</sup>	Reprojection error:	1.15 pix

Camera information				
Camera model	Resolution	Focal length	Pixel size	Pre-calibrated
ILCE-7R	7360 x 4144	Unknown	4.76 x 4.76 μm	No

Figure 5. Camera information, locations and image overlap



**ILCE-7R**

85 images

<b>Resolution</b>	<b>Focal Length</b>	<b>Pixel Size</b>	<b>Precalibrated</b>
<b>7360 x 4144</b>	<b>unknown</b>	<b>4.76 x 4.76 <math>\mu\text{m}</math></b>	<b>No</b>
Type:	Frame	F:	3230.06
Cx:	0	B1:	-7.69146
Cy:	0	B2:	0
K1:	-0.00253342	P1:	0.000594891
K2:	-0.000199392	P2:	-0.000865084
K3:	0	P3:	0
K4:	0	P4:	0

Figure 6. Image residual distribution.

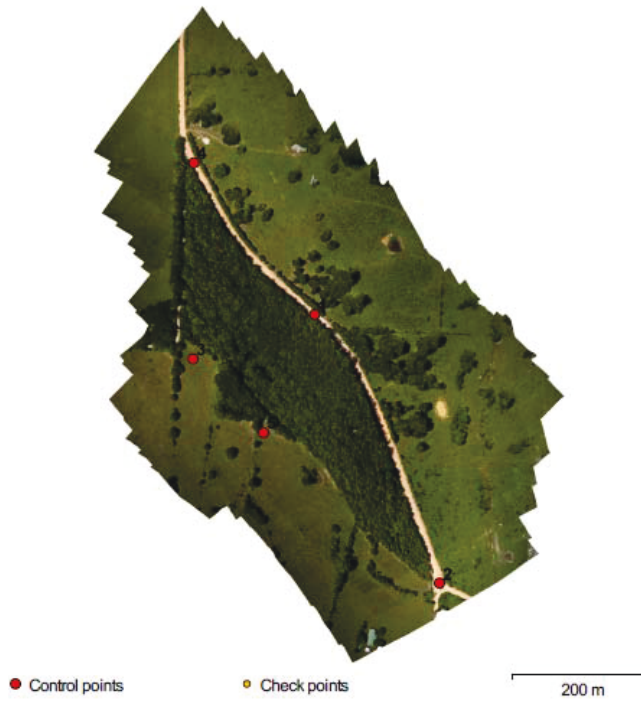


Figure 7. GCPs location

Table 1. Control points

RMSE of each selected control points					
Label	X error (cm)	Y error (cm)	Z error (cm)	Total (cm)	Image (pix)
1	-7.98183	2.69812	-1.64752	8.58509	0.266
2	0.33093	-4.68569	-6.44085	7.97109	0.546
3	-0.196367	-2.77745	-5.70547	6.34864	0.245
4	1.6136	0.608453	1.66136	2.39459	0.127
5	8.00002	2.9977	7.79079	11.5621	0.406
<b>Total</b>	<b>5.10784</b>	<b>3.04323</b>	<b>5.29543</b>	<b>7.96196</b>	<b>0.324</b>

Through image alignment and interpolation, the dense point cloud of the canopy model can be generated. The dense point cloud can be converted to polygonal mesh or triangle mesh models (TIN). The 3D dense point cloud of the study site draped with aerial photos is displayed in the Figure 8. And the 3D TIN of the study site is illustrated in Figure 9.

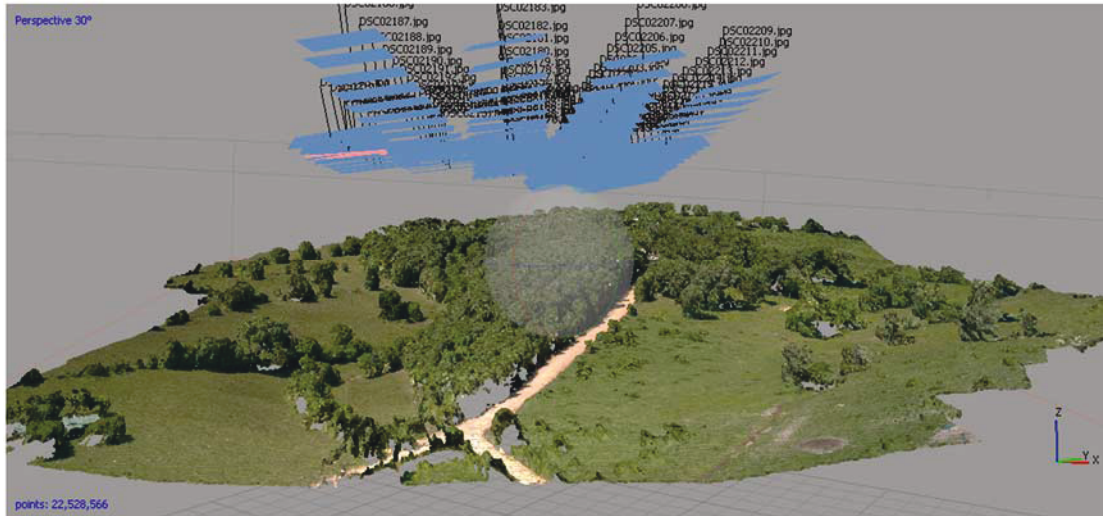


Figure 8. 3D dense point cloud of the study site

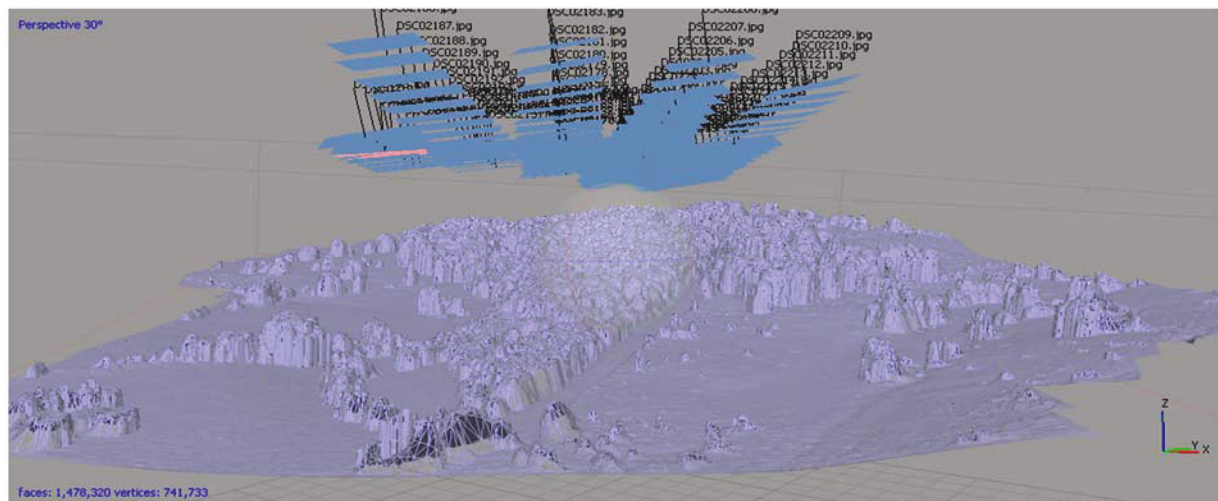


Figure 9. 3D Polygonal mesh of the study site

Based on the 3D dense point cloud and Polygonal mesh, the software can generate a mosaicked orthophoto as the texture for the 3D canopy model, as shown in Figure 10.

**Generate Digital Elevation Model (DSM).** The DSM in Figure 11 can be generated based on the dense cloud or mesh model. During DSM generation progress, the coordinate system should be specified in accordance with the system used for the model referencing.

The DSM must be export to the Geotiff format in order to be used in other software. The Geotiff format is a metadata standard that allows geographic information embedded in TIFF files. The information includes map projections, coordinate systems, ellipsoids, and datum. GeoTIFF format is fully compliant with TIFF 6.0, so the GeoTIFF format files can still be opened by the software which is not able to read and interpret specialized metadata. The exported Geotiff image is displayed as in the Figure 12.



Figure 10. Texture or orthophoto of the study site.

There are several ways to visualize the DSM generated from PhotoScan. In this study, I use ArcScene 10.4 to display a 3D visualization scene when setting a base height of the image on the surface model. And then an image segmentation method can be used to recognize and classify individual trees effectively (Figure 13).

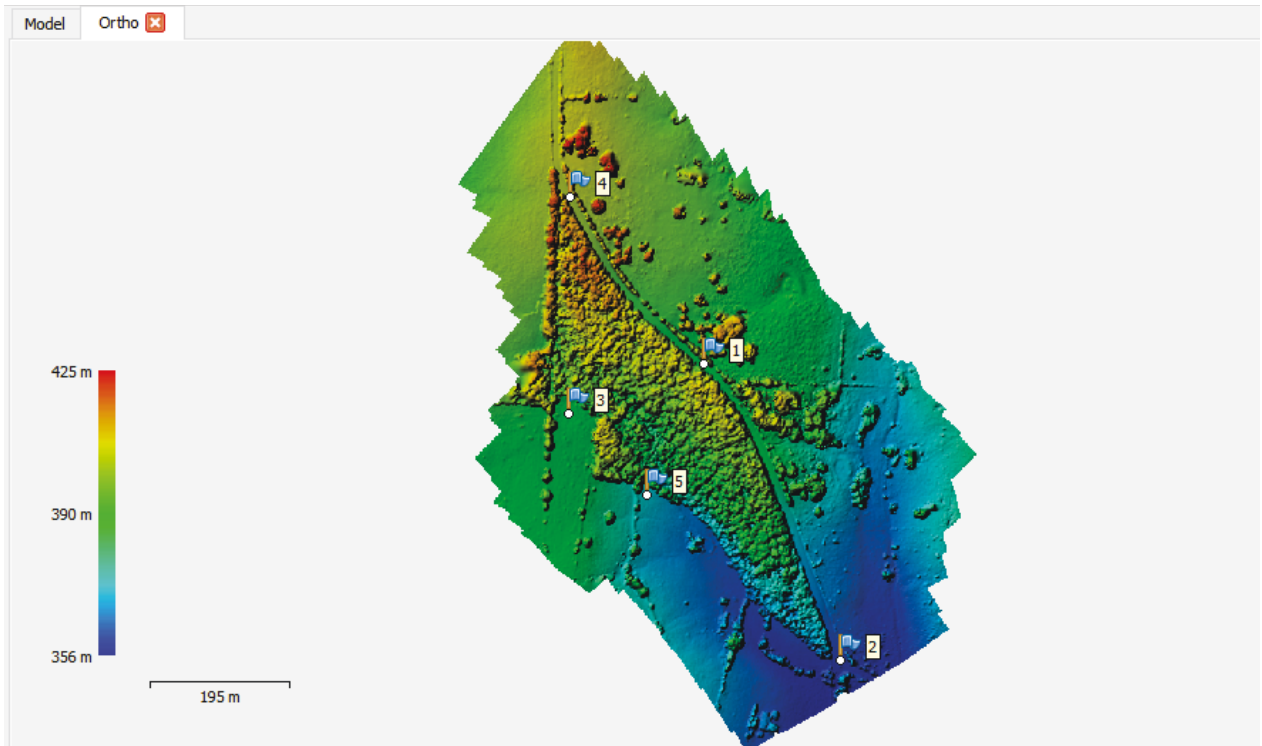


Figure 11. DSM generated from dense cloud. Resolution is 15.1 cm/pix, and Point density is 44 points/m<sup>2</sup>.

The Appendix A shows all the processing parameters of the photogrammetric software. It included the point clouds, DSM and reconstruction parameters.



Figure 12. The exported DSM to geotiff format

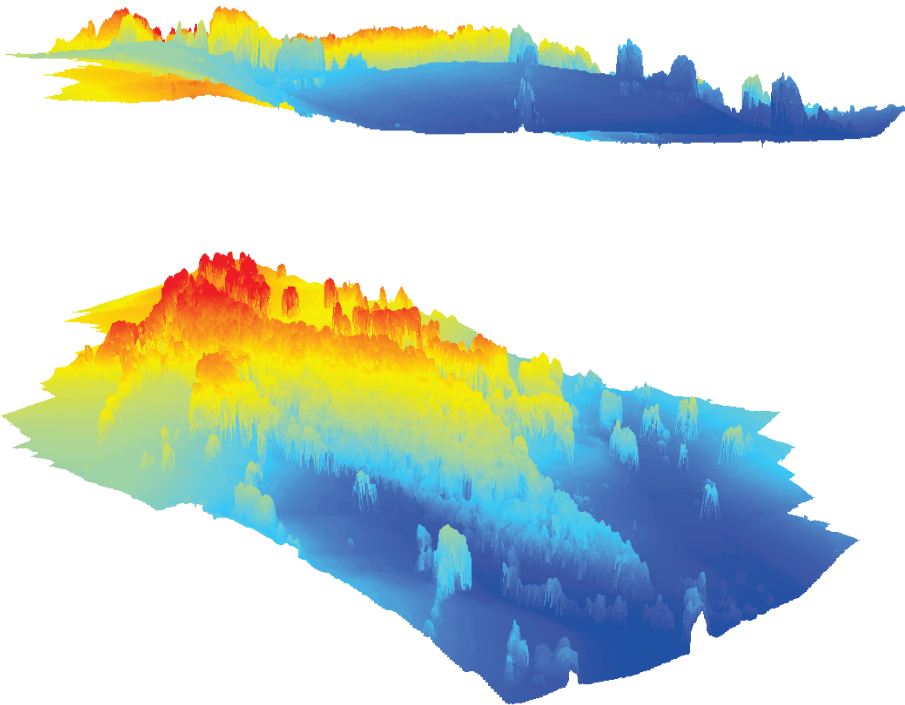


Figure 13. 3D visualization of DSM

## Image Segmentation and Classification

I decide to use Python as the programming language for image processing and segmentation. As an open source package, it allow users to different image processing modules, and supported by numerical and scientific libraries and code optimization toolboxes (Caeiro *et al.*, 2014).

Python version 3.7 is the newest version which has been used in this research. Before proceeding to the image processing stages, the required libraries have been successfully installed such as gdal, numpy, matplotlib, pillow and some supporting modules such as ImagePy, Scikit-image and OpenCV.

**Libraries Installation.** The first step in image processing is how the system and programming software can read DSM's digital format. scikit-image which is a library module includes image processing features as important as the filters, the processing algorithm image segmentation; rasterio a library module for the purpose of processing the image dataset in the form of numbers, the one-dimensional array, two-dimensional or multidimensional. With powerful capabilities of an open-source software, users can gain access to a massive treasure shared library for free. Here, the modules and libraries like GDAL (Geospatial Data Abstraction Library) (GDAL Development Team, 2018) version 2.2.4, OpenCV, numPy, Scikit-image, RasterIO has been installed.

The modules GDAL, OpenCV and Scikit-Image shall help the researcher by reading the DSM image into the memory as a floating-point array, then they can process the data as numeric data and after that display the results as images. With the numeric data, the researcher can apply different image filtering by using filtering modules which are integrated in OpenCV and Scikit-



image in order to get better result when applying different filtering methods to segment the image and then find locations of trees.

**Image Segmentation and Classification.** The image segmentation is an important topic in computer vision and image processing field. Image segmentation is the process of image processing by dividing the image into the small objects similar to a set of local pixel representation. The goal of segmentation is to simplify and separate the image into a number of objects which are more meaningful and easier to identify comparing to the real world (Sonka *et al.*, 1999).

In the past, there were some researches have been done before with several methods for tree detection and segmentation. In case there is no overlapping parts of trees on the image or canopy surface model, trees are separable by thresholding. This algorithm is convenient and efficient to segment objects in an image or DSM. According to those researches, obtaining individual tree information requires that trees are able to be visually delineated in the remotely sensed data (Strîmbus, 2015). Most of those methods have used object-oriented image analysis methods such as image segmentation (Laliberte *et al.*, 2004; Laliberte *et al.*, 2007), multi-resolution segmentation, feature extraction, object identification like Spatial Wavelet Analysis (SWA) (Falkowski *et al.*, 2006) and object-oriented classification.

However, among these studies, the image data is used primarily obtained from LiDAR. For example, the height of the coniferous trees and crowns diameter can be estimated through two-dimensional spatial wavelet approach for LiDAR (Falkowski *et al.*, 2006). And single-tree identification is also realized for LiDAR data to simulate the annually increase of current stem volume (Bottai *et al.*, 2013). A segmentation based on graph is developed to extract trees from airborne LiDAR data (Strîmbus, 2015). And a method to determine the location of each tree is

developed by using local maximum detection (Pouliot *et al.*, 2002). Alternatively, I develop a method to extract each tree from the photogrammetry-based DSM in this study.

Moreover, in practice there are many instances where the same objection on the surface convex, concave or overlap, not separately such as the digital surface images of vegetation areas. In these cases, the thresholding and contour detection algorithm may not work accurately then the other image segmentation algorithms will be applied together.

The watershed segmentation is chosen as the foundation of this algorithm. The watershed segmentation will detect all local minima on the DSM, which sometimes generates an overwhelming number of small segments (Zhao and Popescu, 2007). However, this algorithm is successfully applied for Developing Efficient Procedures for Automated Sinkhole Extraction from Lidar DEMs (Miao *et al.*, 2013), where the DEMs shown the sinkholes as concave areas. Furthermore, some other methods like hierarchical watershed transform (HWT) algorithm to segment the canopy height model (CHM) which is obtained by examining the “dynamic” properties of local maxima (Zhao and Popescu, 2007).

In this study, the watershed segmentation has been applied assuming the 3D canopy model is an upside-down watershed. The detection of local maxima has been chosen instead of local minima. The canopy height model (CHM), in theory can be identified by determining the difference between DSM and bare ground digital terrain model (DTM):

$$\text{CHM} = \text{DSM} - \text{DTM}$$

However, in this study, the DTM data of the study site has not been found, therefore, the DTM is assumed to be a relatively flat surface with the average altitude is zero, or in this case the CHM is equivalent to the DSM. From now on in this study, the DSM will be used as the CHM in every process related to the canopy height. Figure 14 shows the CHM of the study site.

The watershed segmentation algorithm has been studied for transformation and rewritten in the Python programming language, which will instructs the computer to read all the pixel values (in essence, the altitude values) as ndarray package (Hoyer and Hamman, 2017) that can read them uses rasterio (Gillies *et al.*, 2013). It will provide multidimensional arrays and datasets with metadata (Annala, 2018) and then merge them together according to the segmentation to determine positive areas that identify individual tree. In Python, the Scikit-image (van der Walt., 2014; Pereira *et al.*, 2015), GDAL and OpenCV are used for processing numeric data contained in the DSM raster (Pedregosa *et al.*, 2011).

In this study, the watershed segmentation algorithm has been applied with two different filtering techniques, i.e. mean shift filter with adaptive thresholding and Gaussian filter.

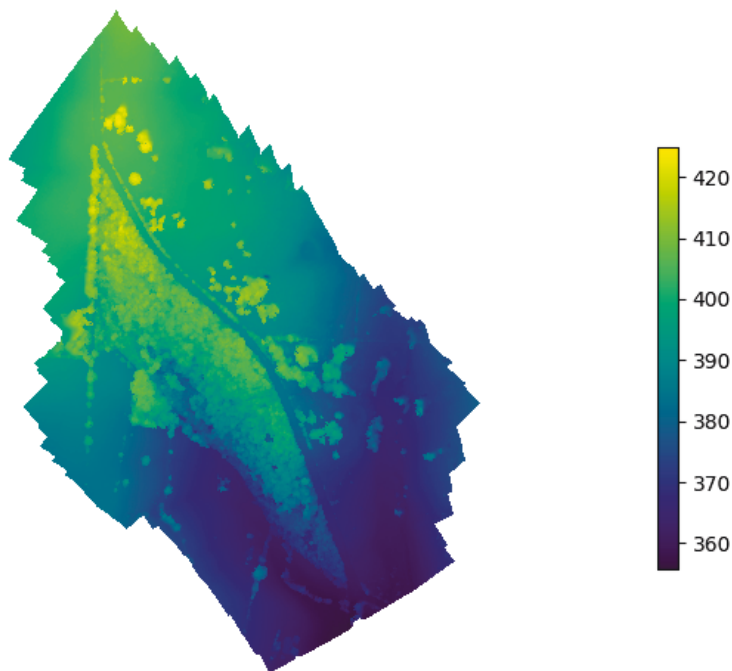


Figure 14. Canopy height model of the study site

Watershed Segmentation with Mean Shift Filtering and Adaptive Thresholding. In the method which using mean shift filter and adaptive thresholding, the DSM image has been imported by using reading function of OpenCV. It will return the DSM image as a numpy array. After imported, due to the limitation of the filter which only support 8-bit imagery, the imported DSM image has been converted to 8-bit image by using `gdal.translate` function.

The mean shift filtering algorithm is a data clustering algorithm that is commonly used in the computer vision and image processing. One of the interesting properties of this filter is that it can largely preserve the edges so as to enhance the image segmentation.

Syntax:

```
cv2.pyrMeanShiftFiltering(src, sp, sr[, dst[, maxLevel[,
termcrit]])
```

where *src* is source image (8-bit only), *sp* is the spatial window radius, *sr* is the color window radius, *maxLevel* is the maximum level of the pyramid for the segmentation and *termcrit* is termination criteria when stop meanshift iterations.

At each pixel (X, Y) of the input image, the function performed iterations mean, mean neighborhood pixel (X, Y) in space common space to be considered:

$$(x, y) : X - sp \leq x \leq X + sp, Y - sp \leq y \leq Y + sp$$

In the surrounding area of the average value of space (X', Y') and average vector is found, and it act as neighborhood centers on the next iteration:

$$(X, Y) \rightarrow (X', Y')$$

After repetition, the value of the original pixels which are the points from where the repetition started, are set to the last values. Whenever  $maxLevel > 0$ , firstly gaussian pyramids of varying  $maxLevel + 1$  was built and on the process to be run on smallest layer. Then, the results

are transmitted to the larger layer and the repetitions are only running back on the pixels in which the color of different layers more than  $sr$  compared with lower resolution layer of the pyramid. That makes the boundary of the region. Note that the result would be different than the results obtained by running the mean shift procedure on the entire original image (i.e. when  $maxLevel == 0$ ) (“OpenCV development team”, n.d.).

In the next step, the approximate estimate of the trees can be found by using image binarization of adaptive Image thresholding algorithm (Yousefi, 2011). Aiming to smoothen image before applying thresholding, the mean-shift filter has been applied and the original image has been converted into grayscale.

Using the adaptive thresholding in Python, the code would be defined as in the following:

```
1. gray = cv2.cvtColor(shifted, cv2.COLOR_BGR2GRAY)
2. thresholding = cv2.threshold(gray, 255,
    cv2.ADAPTIVE_THRES_MEAN_C, cv2.THRESH_BINARY, 25, 5)
3. cv2.imshow("Thresholding", thresholding)
```

Then, in the Figure 15, viewers can see the result of the adaptive thresholding application.

For segmentation of trees in a digital surface model image, the watershed algorithm will be applied by computing the Euclidean Transform Distance to the closest zero (usually background pixel’s value) for each of the foreground pixels by using:

```
Distance = ndimage.distance_transform_edt(thresh)
```

After that, finding the local maxima (when using this algorithm for extracting the trees, the local maxima will be used instead of local minima as in watershed) by using function `peak_local_max()` which has been integrated in the module `scikit-image`.

Syntax:

```
skimage.feature.peak_local_max(image, min_distance=1,  
threshold_abs=None, threshold_rel=None, exclude_border=True,  
indices=True, num_peaks=inf, footprint=None, labels=None,  
num_peaks_per_label=inf)
```

The `peak_local_max` routine will find peaks in an image and render output as coordinate list or boolean mask. Those peaks are understood as the local maxima in a region of  $2*min\_distance+1$  (in this case, it is considered that those peaks are separated by at least  $min\_distance$ ). In case multiple local maxima are found with identical pixels within the defined region, the output will return all coordinates of those pixels.



Figure 15. Result of thresholding

On the other hand, in case providing the minimum intensity of peaks, the maximum of the two is chosen as the minimum intensity threshold of them (“Scikit-image development

team”, n.d.). The local maxima can be found on the Canopy Height Model (CHM) where the local maxima is considered equivalent to the top of the tree (Figure 17).

```
localMax = peak_local_max(image, indices=False, min_distance=10, labels=thresholding)
```

Then apply the watershed function:

```
Markers = cv.watershed (image, markers)
```

where *image* is used as input array and *markers* is used as output array. The code for applying watershed segmentation algorithm:

```
1. Dist = ndimage.distance_transform_edt(thresh)
2. localMax = peak_local_max(image, indices=False, min_distance=10, labels=thresholding)
3. markers = ndimage.label(localMax, structure=np.ones((3, 3)))[0]
4. labels = watershed(Dist, markers, mask=thresh)
```

Then the watershed function will return the result as an array which has the same dimension of the original image and each pixel has a unique value. It will be known as the pixel which has the same value will be in the same object.

Finally, given the contour of the objects by drawing a boundary surrounding each object which has been identified before and display them on to output screen as shown in Figure 16. In order to visualize those detected objects, a loop over unique values procedure is being done by for-loop function. The complete code of Python programming of this method is shown in Appendix B.

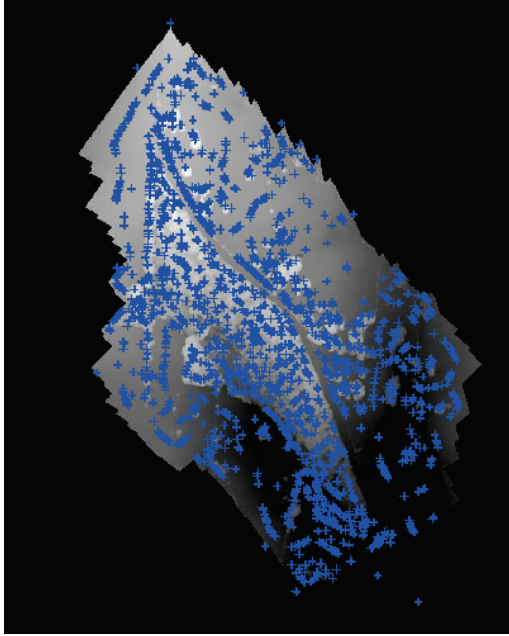


Figure 16. Result of segmentation when using mean shift filtering

Watershed Segmentation with Gaussian Filtering. The remaining method of image filtering is to apply Gaussian filter before handling the watershed segmentation procedure. During the study period, I found that the identification of the point with maximum height of each zone will be affected to the accuracy level when the computer even has recognized the low shrub instead of only high trees. Therefore, surface smoothing help researcher eliminate this point.

In order to get the best result, the parameters of the Gaussian filter are adjusted gradually until the results are close to the reality while visually compared with the original aerial photographs taken at the study site.

Gaussian filtering basically is a kernel defined as

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



where  $x, y$  is the distance from the original point in the horizontal and vertical axis respectively and  $\sigma$  is the standard deviation of the Gaussian distribution. When applied this formula in two dimensions, it creates a surface that contours are concentric circles with Gaussian distribution from the center point. The value of this distribution is used to build convolution matrix is applied to the original image (Shapiro, L. G. and Stockman, G. C., 2000).

The new value of each pixel is set to weighted average number of pixels of that neighborhood. The original pixel values received the heaviest weight (Gaussian highest value) and the neighboring pixels get smaller weights. By this way, it will smooth out many unnecessary local variations on the CHM, so that larger tree's boundaries will be enhanced.

Syntax:

```
scipy.ndimage.gaussian_filter(input, sigma, order=0, output=None,
mode='reflect', cval=0.0, truncate=4.0)
```

where *input* is the array file which was imported by using rasterio and the important sigma parameter which decide the smoothness of filtering process. After applying the Gaussian filter and peak local maxima process, the result has been shown in the Figure 17.

After that, apply the same procedures of the watershed segmentation to get the image to be segmented. The result of image segmentation in case of using Gaussian filter was displayed as an image in Figure 18 and scattered as 3D model in Figure 19a. By visually examining with the help of original aerial photos and mosaicked orthophotos, the tree extracted from Gaussian-based watershed segmentation is significantly more accurate comparing to mean-shift filtering watershed segmentation. The Appendix C included all the codes of this method.

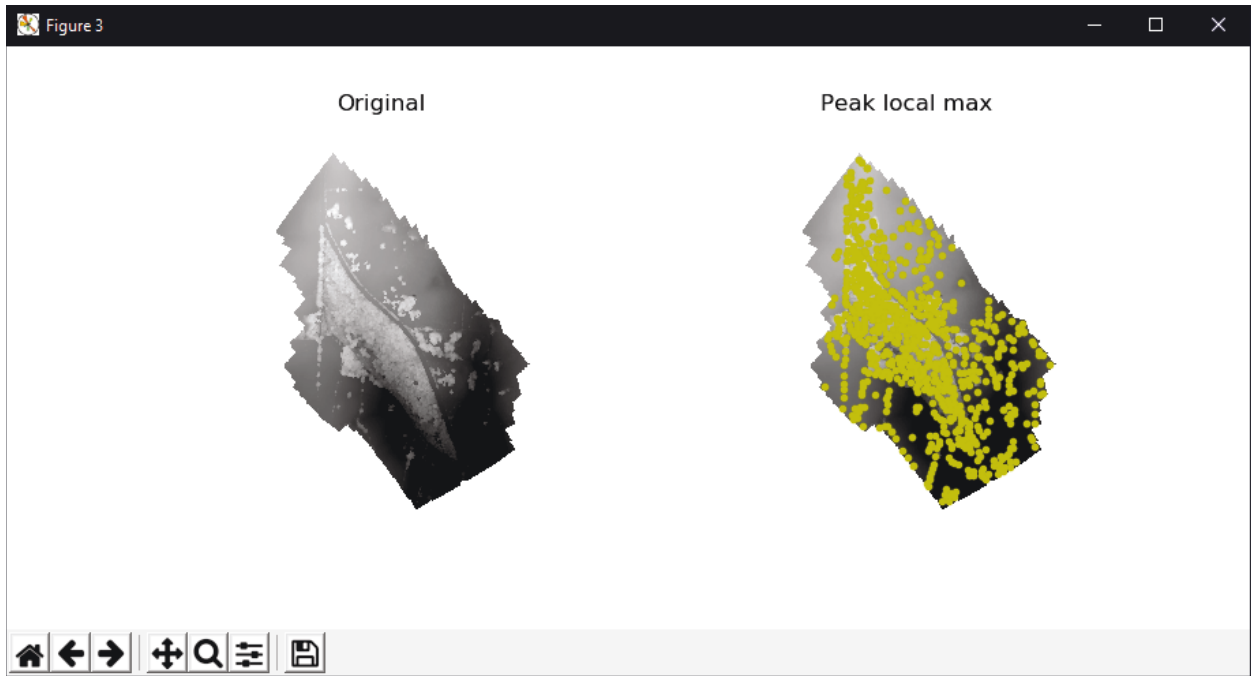


Figure 17. Result of local maxima classification after applying Gaussian filter

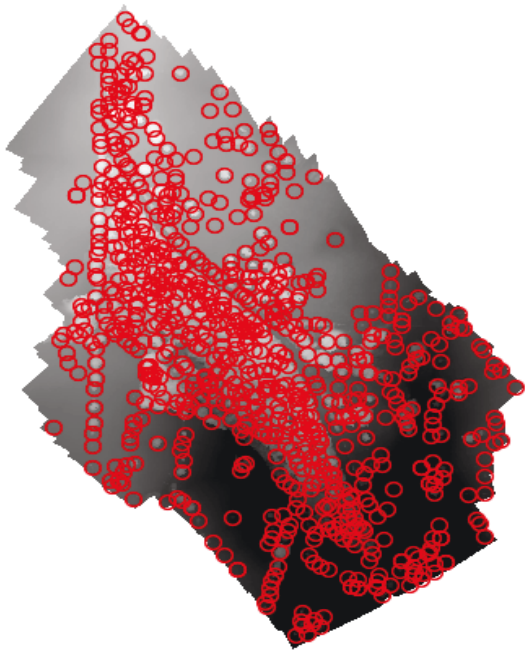


Figure 18. Image segmentation result

The tree height is calculated based on the tree distribution from Gaussian watershed segmentation. The top of trees are considered as the local maxima which have been found, the coordinates of those local maxima points are extracted through peak\_local\_max function. By using the value of these coordinates, the pixel value on the original DSM are extracted. Since there is no database about the DTM, level ground is assumed, and the tree height results are obtained by subtracting the ground elevation values from DSM values. The result can be seen in Figure 19b. The code of this method is in the Appendix D.

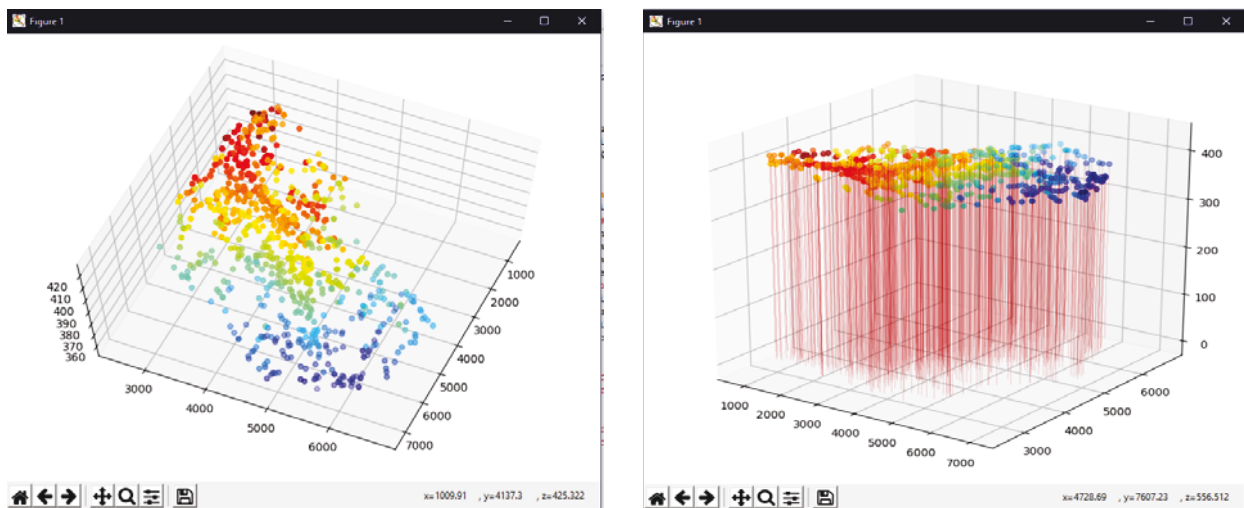


Figure 19. a. 3D model of local maxima; b. 3D model of trees height

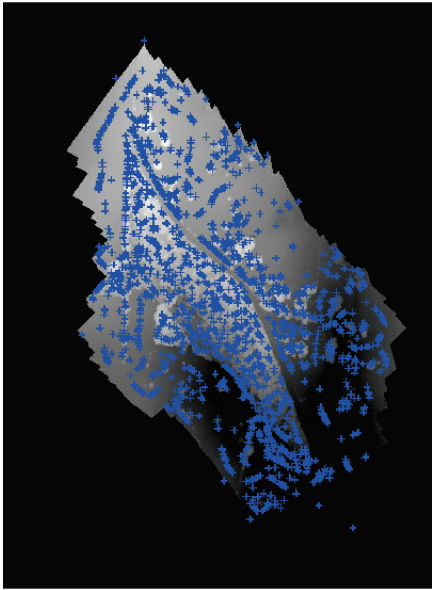
## RESULTS AND DISCUSSION

Based on the results obtained from the two cases have applied different filters which are mean-shift filter and Gaussian filter, the results obtained have been listed in Table 2. The Gaussian filter is more effective and accurate than mean-shift filter in identifying trees by visually examining the results with aerial photos, mosaiced orthophoto and ground reference data (Figure 20, 21).

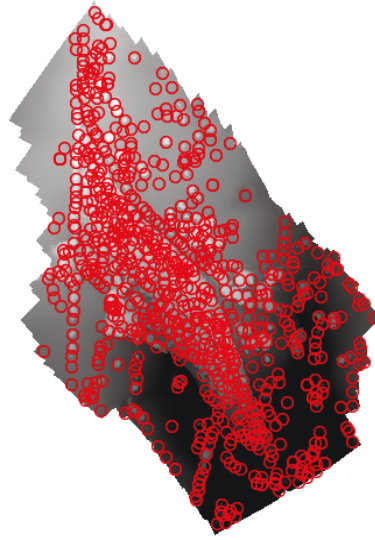
Table 2. Segmentation results

Method	Number of segments (trees) recognized
	Number
With mean-shift filtering	1676
With Gaussian filtering	775

There are several limitations of applying DSM watershed segmentation algorithm. The main challenge lies in that it is difficult to determine the true points of local maxima. DSM is not precise enough to separated clearly between neighboring trees if the canopy is dense. When two or more trees are too close the algorithm can only identify one tree (Figure 22A). Sometimes in the same tree there are two widespread branches with different heights, the algorithm can mistakenly identify two individual trees (Figure 22C).

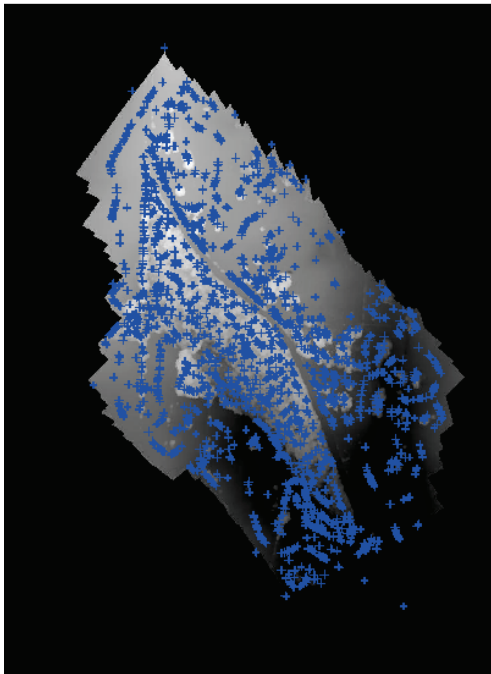


A

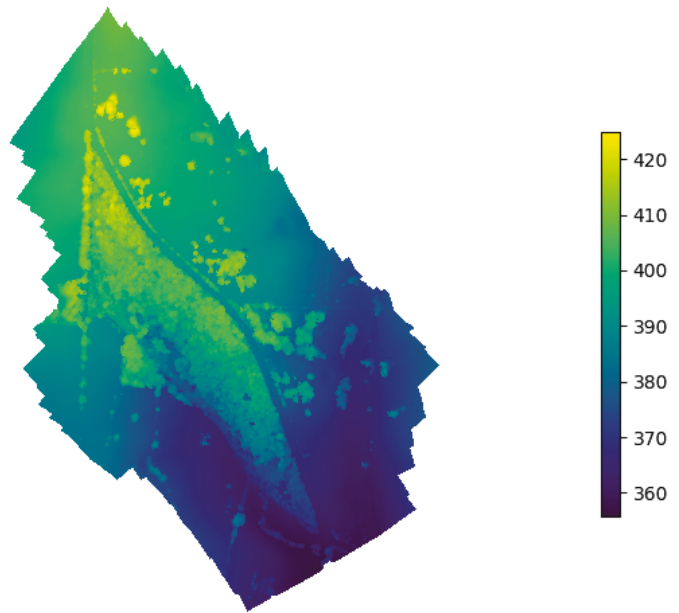


B

Figure 20. Results from two pre-processing filters  
 A. Mean shift filter      B. Gaussian filter



A



B

Figure 21. Result when applying mean shift filter (A); and the Canopy Height model (B)

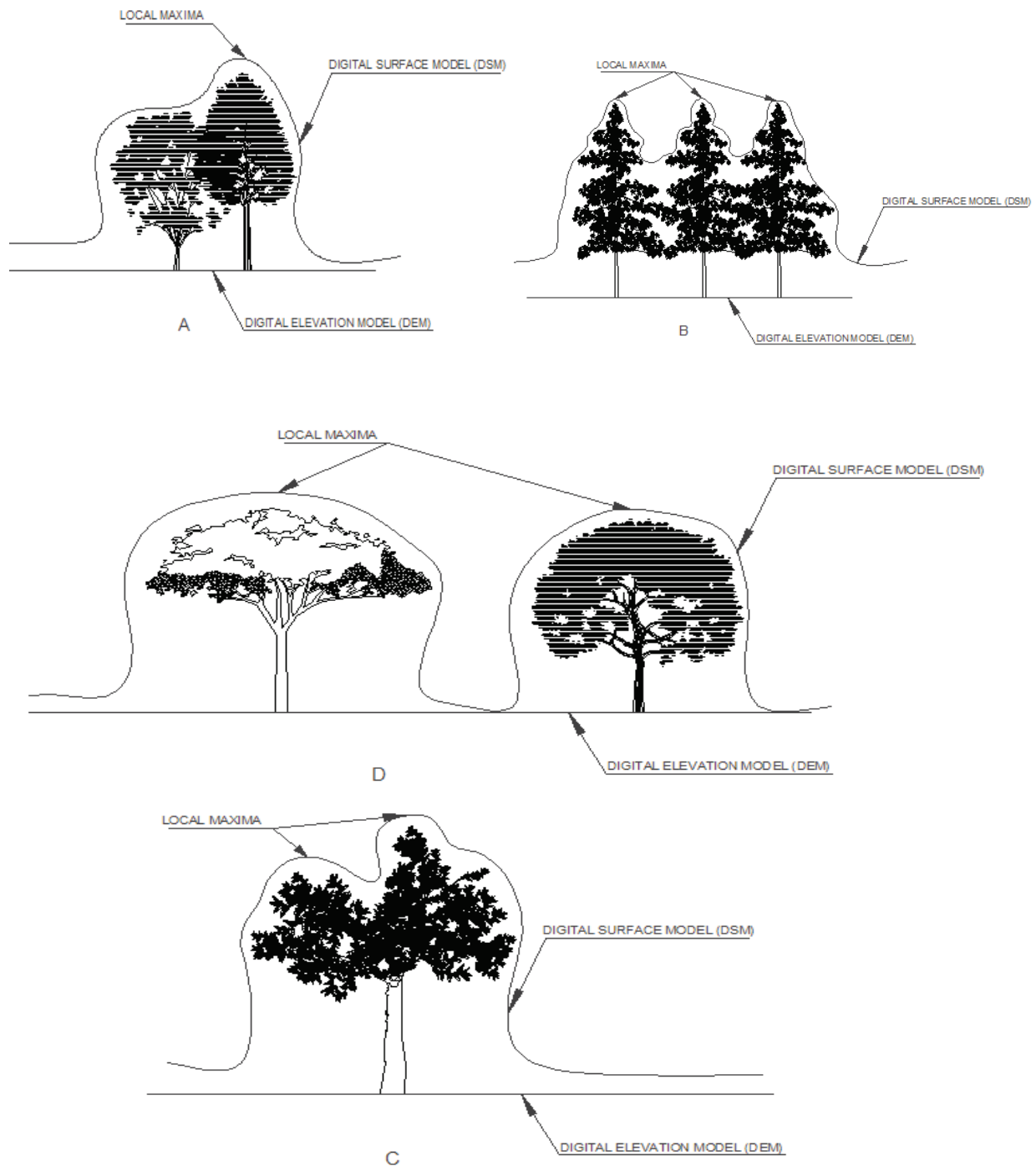


Figure 22. A – Two plants too close then only one being recognized; B – Three separately plants then they can be recognized accurately; C – One tree but with two different widespread elevation then to be recognized as two; D – Two individual plants then they can be recognized accurately.

In order to provide an accuracy statistic about the segmentation results, the Root Mean Square Error (RMSE) statistic method is implemented. Because of the actual quantity of trees on the study site is not provided and cannot be counted at the site, there are thirty individual trees has been chosen on the DSM to perform the statistic base on the pixel values on the images.

The Root Mean Square Error is calculated by:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_1^N (h_{0(x,y)} - h_{(x,y)})^2}$$

Where:

- N is the sample size, N = 30
- $h_{0(x,y)}$  is the pixel value on the DSM
- $h_{(x,y)}$  is the pixel value with the coordinates (x,y) on the filtered image

Figure 23 is the error graph of the Sum of Square Error and the RMSE result is shown in the Table 3. They show that the mean-shift filter image has a large deviation from the DSM when the Gaussian filter image has a low deviation from the DSM. This proves that the segmentation result using mean-shift filter has larger error, quite similar to the result which is shown in the images.

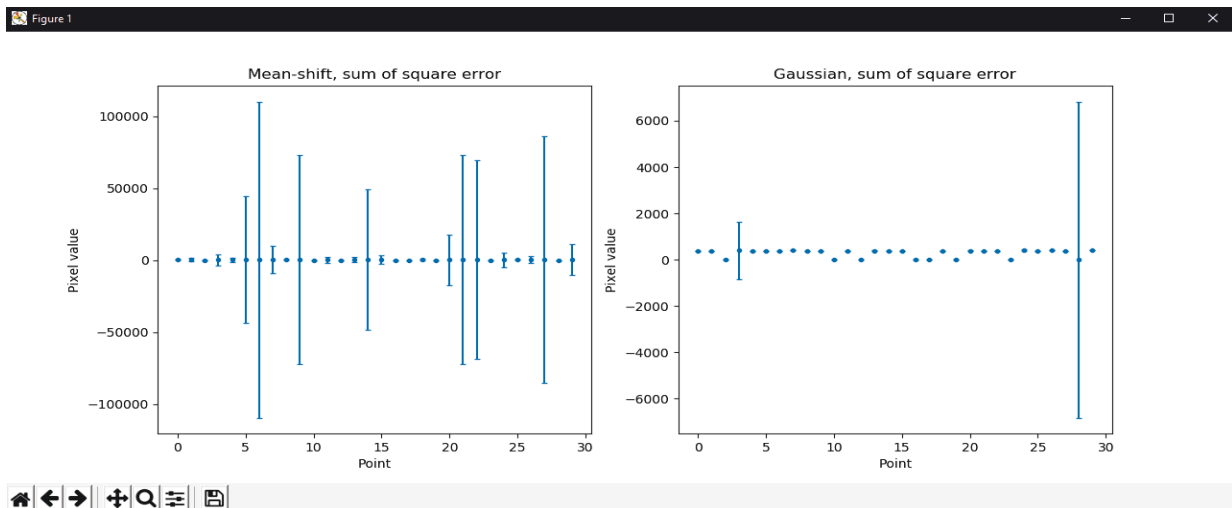


Figure 23. Sum of Square Error graph

Table 3. RMSE result of the selected trees

Point	Sum of Square Error			
	X	Y	SSE Mean-shift	SSE Gaussian
1	4400	2749	16.6726	0.1863
2	3510	5096	987.2629	0.0014
3	4119	2349	0.000	0.1983
4	3807	2660	3,897.8307	1,240.3818
5	4505	3032	1,460.4981	0.0002
6	5440	4650	44,035.4136	0.1755
7	6094	5007	109,814.5252	0.0050
8	2961	2794	9,394.1762	0.0046
9	2931	4739	96.9264	0.0061
10	4549	5868	72,602.2781	0.0000
11	3391	2289	0.000	0.000
12	4386	3284	2,042.9759	0.0001
13	3575	2066	0.000	0.000
14	4169	3299	1,835.8797	9.4503
15	5431	4650	48,996.3565	1.1732
16	5416	5066	2,966.8317	2.8147
17	3382	2274	0.000	0.000
18	3486	1963	0.000	0.000
19	4303	3121	489.4793	0.0085
20	3456	1859	0.000	0.000
21	5416	4234	17,537.2958	17.5914
22	5075	4116	72,653.7790	0.0054
23	5550	4012	69,024.2689	0.0006
24	3219	1695	0.000	0.000
25	4778	3596	5,058.9543	36.3215
26	3486	4041	8.7788	0.2045
27	3560	2779	2,168.9752	0.0008
28	5194	4368	85,534.7314	3.5016
29	2951	2482	0.000	6,824.5219
30	2951	2601	10,770.3037	0.0003
<b>Root Mean Square Error of Mean-Shift filter:</b>			<b>136.79598</b>	
<b>Root Mean Square Error of Gaussian filter:</b>			<b>16.46871</b>	



## CONCLUSIONS

The motivation of the study is to reconstruct 3D digital surface model from the captured images from unmanned aerial system through photogrammetry, and then extract individual trees through image segmentation approach. As a modern photogrammetric software, Agisoft PhotoScan is provide an excellent result by identifying tie points efficiently, and further generate dense point clouds and DSM. The result of 3D image reconstruction is evaluated as successful when the 3D scene of the study site has been reconstructed comparing to GCPs.

The watershed segmentation is tested with mean-shift filter and Gaussian filters to extract individual trees from the derived DSM. Watershed segmentation algorithm has simplified memory access, least computational complexity and good segmentation results compared to the other image segmentation algorithms. Segmentation results are visually acceptable when using Gaussian filter. However, the performance of the mean-shift filter is not satisfactory.

There is still an over-segmentation phenomenon after the pre-processing and segmentation steps, and hence a post processing step is necessary to remove the over-segmentations. The implemented watershed segmentation algorithm has the potential to use for different studies on monitoring various types of natural and agricultural vegetation area when using the data from aerial photos, which is much easier and cheaper than using the data from LiDAR technology.

In this study, due to conditions and limitations which were not allowing to involve in measurement at the study site, inspection and measurement on the images collected, so this study cannot provide more detailed accuracy assessment in terms of tree counting. However, the method and open-source Python codes provided in the study shows the potential to accelerate the

time-consuming labor work and will benefit scientists in surveying and monitoring the growth rate of the woodland or forests.

## REFERENCES

- AgiSoft, 2017. PhotoScan Professional (Software). Available at <http://www.agisoft.com/downloads/installer/>
- Annala, L., M. Eskelinen, J. Hämäläinen, A. Riihinen, and I. Pölönen, 2018. Practical approach for hyperspectral image processing in Python. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-3, pp.45-52.
- Bertram, T., T. Bock, A. Bulgakov, and A. Evgenov, 2014. Generation the 3D Model Building by Using the Quadcopter. *Proceedings of the 31st International Symposium on Automation and Robotics in Construction and Mining (ISARC)*.
- Bottai, L., L. Arcidiaco, M. Chiesi, F. Maselli, 2013. Application of a single-tree identification algorithm to LiDAR data for the simulation of stem volume current annual increment - *J. Appl. Remote Sens.*, 7, p. 073699
- Caeiro, J., M. Piedade, and E. Ataíde, 2014. Image Processing and a Python-GIMP Based Algorithm Development Environment.
- Falkowski, M.J., A.M.S. Smith, A.T. Hudak, P.E. Gessler, L.A. Vierling, and N.L. Crookston, 2006. Automated estimation of individual conifer tree height and crown diameter via two-dimensional spatial wavelet analysis of LiDAR data. *Canadian Journal of Remote sensing*. 32(2):153-161.
- GDAL Development Team, 2018. Gdal - geospatial data abstraction library, version 2.2.3. Official website: <http://www.gdal.org>
- Gillies, S. *et al.*, 2013—. Rasterio: geospatial raster i/o for Python programmers. Source code available at <https://github.com/mapbox/rasterio>.
- Gupta, S. and D. Shukla, 2018. Application of drone for landslide mapping, dimension estimation and its 3D reconstruction. *Journal of the Indian Society of Remote Sensing*.
- Hoyer, S. and J. Hamman, 2017. Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*. Source code available at <https://github.com/pydata/xarray>.
- Iovan, C., D. Boldo, and M. Cord, 2008. Detection, segmentation and characterization of vegetation in high-resolution aerial images for 3D city modelling, *The International Archives of the Photogrammetry, Remote sensing and Spatial information Science*. Vol. XXXVII. Part B2a, Beijing.

- Kim, C., H. Moon, and W. Lee, 2016. Data management framework of drone-based 3D model reconstruction of disaster site. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B4, pp.31-33.
- Laliberte, A.S., A. Rango, K.M. Havstad, J.F. Paris, R.F. Beck, R. McNeely, and A.L. Gonzalez, 2004. Object-oriented image analysis for mapping shrub encroachment from 1937 to 2003 in Southern New Mexico, *Remote sensing of Environment*, 93(1-2):198-210.
- Laliberte, A.S., E.L. Fredrickson, and A. Rango, 2007. Combining decision trees with hierarchical object-oriented image analysis for mapping arid rangelands. *Photogrammetric Engineering and Remote sensing*, 73(2):197-207.
- Miao, X., X. Qiu, S. Wu, J. Luo, D. R. Gouzie, and H. Xie, 2013. Developing Efficient Procedures for Automated Sinkhole Extraction from Lidar DSMs. *Photogrammetric Engineering & Remote Sensing*, 79(6), 545-554. doi:10.14358/pers.79.6.545
- OpenCV development team. (n.d.). Retrieved from <https://docs.opencv.org/3.0-beta/modules/imgproc/doc/filtering.html>
- Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Pereira, P. J., H. Weinacker, B. Koch, and M. B. Schimanski, 2015. Scikit-image for trees local maxima detection.
- Pouliot, D.A., D.J. King, F.W. Bell, and D.G. Pitt, 2002. Automated tree crown detection and delineation in high-resolution digital camera imagery of coniferous forest regeneration. *Remote Sens. Environ.* 82,322-334.
- Scikit-image development team (n.d.). Retrieved from [https://scikit-image.org/docs/dev/api/skimimage.feature.html#skimimage.feature.peak\\_local\\_max](https://scikit-image.org/docs/dev/api/skimimage.feature.html#skimimage.feature.peak_local_max)
- Sebesta R. W., 2002. Concepts of Programming Languages. 5th Edition, Addison-Wesley.
- Shapiro, L. G. and G. C. Stockman, 2000. *Computer Vision*, page 137, 150. Prentice Hall.
- Sonka M., V. Hlavac, and R. Boyle, 1993. *Image Processing, Analysis, and Machine Vision*. PWS Publishing.
- Strîmbu, V. F., and M. B. Strîmbu, 2015. A graph-based segmentation algorithm for tree crown extraction using airborne LiDAR data - *ISPRS Journal of Photogrammetry and Remote Sensing*, ISSN: 0924-2716, Vol: 104, Page: 30-43

- The Scipy community (n.d.). Retrieved from [https://scipy.github.io/devdocs/generated/scipy.ndimage.gaussian\\_filter.html#scipy.ndimage.gaussian\\_filter](https://scipy.github.io/devdocs/generated/scipy.ndimage.gaussian_filter.html#scipy.ndimage.gaussian_filter)
- van der Walt, S., J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikitimage contributors, 2014. *Scikit-image: image processing in Python*. PeerJ 2, pp. e453. Source code available at <https://github.com/scikit-image/scikit-image>.
- Yousefi, J., 2011. *Image Binarization using Otsu Thresholding Algorithm*, University of Guelph, Ontario, Canada
- Zhao, K., and S. Popescu, 2007. Hierarchical watershed segmentation of canopy height model for multi-scale forest inventory, *Proceedings of the ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, 12-14 September, Espoo, Finland, pp.436–441.

## APPENDICES

### Appendix A. Processing Parameters

#### General

Cameras 85

Aligned cameras 85

Markers 5

Coordinate system WGS 84 / UTM zone 15N (EPSG::32615)

#### Point Cloud

Points 82,777 of 89,122

RMS reprojection error 0.263878 (1.15117 pix)

Max reprojection error 3.58127 (53.9042 pix)

Mean key point size 4.2145 pix

Effective overlap 2.99841

#### Alignment parameters

Accuracy High

Pair preselection Generic

Key point limit 40,000

Tie point limit 4,000

Constrain features by mask No

Adaptive camera model fitting Yes

Matching time 8 minutes 55 seconds

Alignment time 1 minutes 27 seconds

### **Optimization parameters**

Parameters f, b1, k1, k2, p1, p2

Optimization time 6 seconds

### **Depth Maps**

Count 85

### **Reconstruction parameters**

Quality Medium

Filtering mode Moderate

Processing time 10 minutes 19 seconds

### **Dense Point Cloud**

Points 22,528,566

### **Reconstruction parameters**

Quality Medium

Depth filtering Moderate

Depth maps generation time 10 minutes 19 seconds

Dense cloud generation time 7 minutes 36 seconds

### **DEM**

Size 8,185 x 8,176

Coordinate system WGS 84 / UTM zone 15N (EPSG::32615)

### **Reconstruction parameters**

Source data Dense cloud

Interpolation Enabled

Processing time 3 minutes 21 seconds

## Software

Version 1.2.6 build 2834

Platform Windows 64 bit

## Appendix B. Source code with mean-shift filtering

```
1. from skimage.feature import peak_local_max
2. from skimage.morphology import watershed
3. from scipy import ndimage
4. import cv2
5. from osgeo import gdal
6. import numpy as np
7. from skimage.feature import peak_local_max
8. from scipy.ndimage import gaussian_filter, distance_transform_edt
9. import matplotlib.pyplot as plt
10. import imutils
11. from skimage import data
12.
13. image = cv2.imread('C:/Users/Duong Ha Hai/Documents/All in US/MSU/Thesis
    /Python code/thesis_new8bit.tif')
14. shifted = cv2.pyrMeanShiftFiltering(image, 2, 20) #/Support 8-
    bit only
15. img = cv2.resize(image, (600, 600))
16. cv2.imshow('Input', img)
17. sht = cv2.resize(shifted, (600, 600))
18. cv2.imshow('Shifted', sht)
19. print('Shift ', shifted.dtype)
```



```

20.
21. gray = cv2.cvtColor(shifted, cv2.COLOR_BGR2GRAY)
22. thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
    cv2.THRESH_BINARY, 25, 5)
23. thr = cv2.resize(thresh, (600, 600))
24. cv2.imshow('Threshold', thr)
25. print('Thres ', thresh.dtype)
26.
27. Dist = distance_transform_edt(thresh)
28. print(Dist.shape)
29. ds = cv2.resize(Dist, (600, 600))
30. cv2.imshow('EDT', ds)
31.
32. localMax = peak_local_max(Dist, indices=False, labels=thresh)
33. coordinates = peak_local_max(image, min_distance=3)
34. print(coordinates)
35.
36. markers = ndimage.label(localMax)[0]
37. labels = watershed(Dist, markers, mask=thresh)
38. print("[INFO] {} unique segments found".format(len(np.unique(labels))-
    1))
39.
40. for label in np.unique(labels):
41.     if label == 0:
42.         continue
43.     mask = np.zeros(gray.shape, dtype="uint8")
44.     mask[labels == label] = 255

```



```

8. from osgeo import gdal
9. import numpy as np
10. import cv2
11. import matplotlib.pyplot as plt
12. from scipy.ndimage import gaussian_filter, median_filter
13. import imutils
14.
15. with rasterio.open('C:/Users/Duong Ha Hai/Documents/All in US/MSU/Thesis
    /Python code/Thesis.tif') as dsm:
16.     dsm_im = dsm.read(1, masked=True)
17.     bounds = plotting_extent(dsm)
18.
19. im_gaus = gaussian_filter(dsm_im, sigma=13.5)
20.
21. fig, ax = plt.subplots(figsize = (8,8))
22. dsm_plot = ax.imshow(dsm_im, cmap='plasma')
23. fig.colorbar(dsm_plot, fraction=.023, ax=ax)
24. ax.set_title("Canopy Height Model - CHM")
25. ax.set_axis_off()
26.
27. thresh = cv2.threshold(im_gaus, 0, 255, cv2.THRESH_BINARY)[1]
28. fig, ax = plt.subplots(figsize = (8,8))
29. ax.imshow(thresh, cmap=plt.cm.gray)
30. ax.autoscale(False)
31. ax.set_title('Thresh')
32. ax.set_axis_off()
33.
34. Dist = ndimage.distance_transform_edt(im_gaus)

```



## Appendix D. Source code of tree height calculation

```
1. import gdal
2. import numpy as np
3. import scipy
4. import scipy.ndimage as ndimage
5. from osgeo import gdal
6. import rasterio
7. from scipy.ndimage import gaussian_filter, median_filter
8. from skimage.feature import peak_local_max
9.
10. with rasterio.open('C:/Users/Duong Ha Hai/Documents/All in US/MSU/Thesis
    /Python code/thesis.tif', 'r') as ds:
11.     arr = ds.read(1, masked=True) # read all raster values
12. with rasterio.open('C:/Users/Duong Ha Hai/Documents/All in US/MSU/Thesis
    /Python code/thesis.tif', 'r') as img:
13.     h = img.read() # read all raster values
14.
15. im_gaus = gaussian_filter(arr, sigma=13.5)
16. coordinates = peak_local_max(im_gaus, min_distance=3)
17.
18. x = 0
19. X = []
20. for i in range(0,coordinates.shape[0]):
21.     x = coordinates[i,0]
22.     X.append(x)
23. print(X)
24.
```

```

25. print('++++')
26.
27. y = 0
28. Y = []
29. for i in range(0,coordinates.shape[0]):
30.     y = coordinates[i,1]
31.     Y.append(y)
32.
33. z = 0
34. Z = []
35. m = 0
36. n = 0
37. for i in range (0,len(X)):
38.     m = X[i]
39.     n = Y[i]
40.     z = h[0,m,n]
41.     Z.append(z)
42.
43. print('The tree is respectively with coordinates: ')
44. Zi = np.zeros(len(X))
45. dx = np.ones(len(X))
46. dy = np.ones(len(Y))
47. dz = Z
48. fig = plt.figure()
49. ax = Axes3D(fig)
50. ax.bar3d(X,Y,Zi, dx, dy, dz, color='r',zsort='max')
51. ax.scatter3D(X,Y,Z,c=Z,cmap=plt.cm.jet)
52. plt.show()

```