Virtual Surveyor based Object Extraction from Airborne LiDAR data

by

Md. Ahsan Habib

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering at
The University of Texas at Arlington
August 2017

Arlington, Texas

Supervising Committee:

Venkat Devarajan, Supervising Professor
Michael T Manry
Jonathan Bredow
Ramtin Madani
Ioannis D Schizas

Acknowledgement

To begin with, I would like to express my deepest gratitude to Professor Venkat Devarajan for his continuous support and encouragement throughout my pursuit of a Doctoral degree. It was a pleasure and honor to work with him. I consider myself extremely fortunate to have an advisor who gave me the full freedom to explore on my own. He always kept me motivated through his wisdom, knowledge, and commitment to work.

I would like to thank my doctoral committee members, Dr. Jonathan Bredow, Dr. Michael Manry, Dr. Ioannis Schizas and Dr. Ramtin Madani who gave me valuable inputs in the form of coursework, in reviewing my research work and making insightful comments.

I would like to thank the National Resources Conservation Service (NRCS) and specifically Mr. Collin McCormick and Mr. Steven Nechero for funding my research work and for providing me with all the raw LiDAR data required for my dissertation. I would like to give special thanks to my colleagues in the Virtual Environment Lab: Dr. George Toscano, Dr. Partha Pratim Acharjee, Dr. Mustafa Alam, Mr. Sree Ram Kamabattula and Mr. Babak Namazi for their scholarly company, support, and useful discussions. I am extremely grateful to all the EE department faculty members and staff, especially Ms. Gail Paniuski, for their various forms of support during my graduate study.

I would also like to thank my family especially my parents for the sacrifices they have made to help me achieve this goal. I sincerely thank my sister Nasreen

for her relentless motivation and support. I would not have been able to finish this work without the love and encouragement from my parents and sister.

Abstract

VIRTUAL SURVEYOR BASED OBJECT EXTRACTION FROM AIRBORNE LIDAR DATA

Md. Ahsan Habib, PhD

The University of Texas at Arlington, 2017

Supervising Professor: Venkat Devarajan

Topographic feature detection of land cover from LiDAR data is important in various fields - city planning, disaster response and prevention, soil conservation, infrastructure or forestry. In recent years, feature classification, compliant with Object-Based Image Analysis (OBIA) methodology has been gaining traction in remote sensing and geographic information science (GIS). In OBIA, the LiDAR image is first divided into meaningful segments called object candidates. This results, in addition to spectral values, in a plethora of new information such as aggregated spectral pixel values, morphology, texture, context as well as topology. Traditional nonparametric segmentation methods rely on segmentations at different scales to produce a hierarchy of semantically significant objects. Properly tuned scale parameters are, therefore, imperative in these methods for successful subsequent classification. Recently, some progress has been made in the development of methods for tuning the parameters for automatic segmentation. However, researchers found that it is very difficult to automatically refine the tuning with respect to each object class present in the scene. Moreover,

due to the relative complexity of real-world objects, the intra-class heterogeneity is very high, which leads to over-segmentation. Therefore, the method fails to deliver correctly many of the new segment features.

In this dissertation, a new hierarchical 3D object segmentation algorithm called Automatic Virtual Surveyor based Object Extracted (AVSOE) is presented. AVSOE segments objects based on their distinct geometric concavity/convexity. This is achieved by strategically mapping the sloping surface, which connects the object to its background. Further analysis produces hierarchical decomposition of objects to its sub-objects at a single scale level. Extensive qualitative and qualitative results are presented to demonstrate the efficacy of this hierarchical segmentation approach.

Table of Contents

List of Illustrations

List of Tables

1. Problem Statement and Motivation

The availability of a massive amount of high-quality LiDAR data gives us an unprecedented opportunity to extract useful and accurate information for a wide variety of applications. The only practical method of analyzing such a huge dataset is through automated algorithms. Feature engineering, i.e. finding a good data representation, is the key to success in interpreting image data by computer (Domingos, 2012). *Hierarchical object representation* offers new possibilities to gain insight into the content of a given digital image. This enables the image classifier to exploit a number of new features. Unlike *object detection*, which detects objects of a certain class (such as buildings, or trees) in a given digital image, *hierarchical object representation* based image analysis aims to address the whole scene (Lang and Blaschke, 2003). Here, the entire image is first segmented in several levels of aggregation to produce its constituent image-objects and sub-objects. Next, an object recognition algorithm is employed to analyze the hierarchical object representation to detect specific objects or a certain object type. Partitioning the given scene into objects is much like the way humans conceptually organize the topographic surface for interpretation (Hay and Castilla, 2008). The section of the brain responsible for analyzing visual information is the visual cortex. There is strong evidence that the visual cortex encodes an object based on a hierarchical model (Riesenhuber and Poggio, 1999). The cortex organizes visual information about a target object into a hierarchy of an increasingly complex object representation. The ability of the human brain to

interpret visual data in an effortless manner and with unparalleled accuracy has inspired designs in automated vision system that model the hierarchical object representation in the visual cortex (Rodríguez-Sánchez et al, 2015).

Automatic separation of the topographic surface and its overlying 3D objects is a challenging problem, especially in complex environments and rough terrain. Arefi, in his 2005 paper, pointed to automated segmentation as one of the major and unsolved problems for interpreting LiDAR data. A decade later, a generic solution to this problem has not been developed yet, as pointed out in a recent survey paper (Cheng, 2016). Nevertheless, extensive research has been conducted in this direction. One classical approach is segmentation through object recognition. However, the presence of a large variety of objects in a scene renders classical segmentation through object recognition ineffective. This is because object-recognition based approaches find image objects using object models, which are known *a priori* and are therefore restricted to a fixed number of objects (whose models were previously stored or learned). Object segmentation of unknown objects with arbitrary shapes in a complex environment poses an important but formidable challenge. Nonparametric segmentation methods can be a solution to this problem. Traditional nonparametric segmentation methods rely on segmentations at different scales to produce a hierarchy of semantically significant objects. Properly tuned scale parameters are, therefore, imperative in these methods for successful subsequent classification. Recently, some progress has been made in the development of methods for the automatic tuning of segmentation scale parameter. However, researchers found that it is very difficult

to automatically refine the tuning with respect to each object class present in the scene. Therefore, the methods fail to correctly deliver the result.

LiDAR-based object detection and extraction work mainly concentrate on protruding man-made features and few very distinguishable natural features such as trees. The most object extraction methods assume that the ground surfaces are usually the lowest feature in a local neighborhood. In some cases, this assumption is wrong. For example, swimming pools, mines, an irrigation channel, construction pit etc. are all man-made objects. Thus, intruding objects are largely left out in the extraction process.

Most natural objects, commonly referred to as landforms, are also ignored. Examples of landform types include hills, mountains, watersheds, dunes, cirques, drumlins, crater, lakes, rivers etc. Maps representing landform distribution in a given terrain is in high demand and is referred to as a geomorphological map. Geomorphological maps are fundamental to many geological quantitative analyses (involving landscape ecology, forestry, and soil science) and to management tools for land use and geomorphological risk (Smith, 2011). Lang et al. (Lang, 2003) argued that the detection of natural objects is comparatively more difficult than the detection of the man-made objects. This is because, of a high degree of homogeneity of a man-made object, the set of heuristics to be used for the recognition process seems to be relatively clear.

To address all these problems, we came up with a novel automated hierarchical 3D object extraction algorithm called Automated Virtual Surveyor based Object Extraction (AVSOE). The research objective of this work was to

develop an algorithm, which can derive hierarchical object representation (automated detection and extraction of intruded and protruded objects and sub-objects – natural and man-made) from the LiDAR image of a given terrain.

2. LiDAR basics

Airborne LiDAR emits a laser pulse from the plane and measures the time it takes to return. This generates a 3D point cloud of the landscape from the time interval, the pulse angle and the absolute location of the sensor. The GPS/IMU carried by airplane provides the position and orientation of the sensor platform. LiDAR collects data at a sampling rate greater than 150 kHz (NOAA Coastal Service Center, 2012). The resultant output is, therefore, a very dense and highly accurate geo-referenced elevation data. The scanning scheme of an airborne LiDAR system is shown in Figure 1.

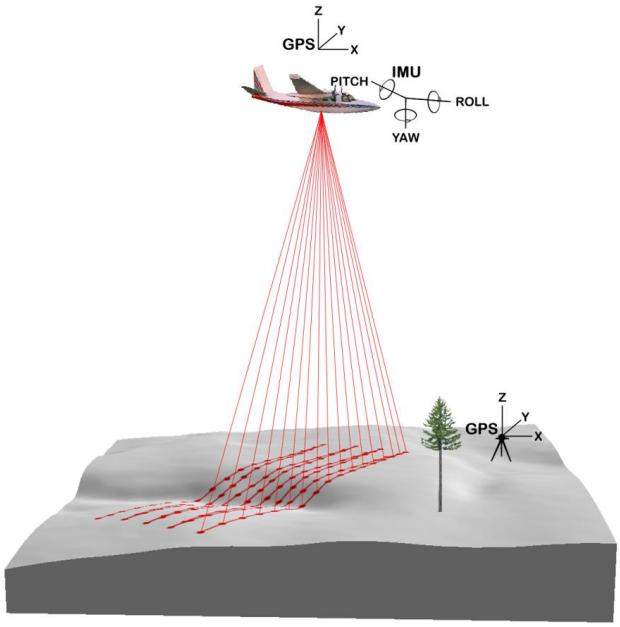


Figure 1 Airborne laser scanning (Monika, 2008)

An important aspect of a LiDAR system is its capability to capture multiple returns per pulse. Figure 2 shows an example, where multiple returns are generated from a single laser pulse. This capability enables accurate characterization of LiDAR returns from the vegetation areas

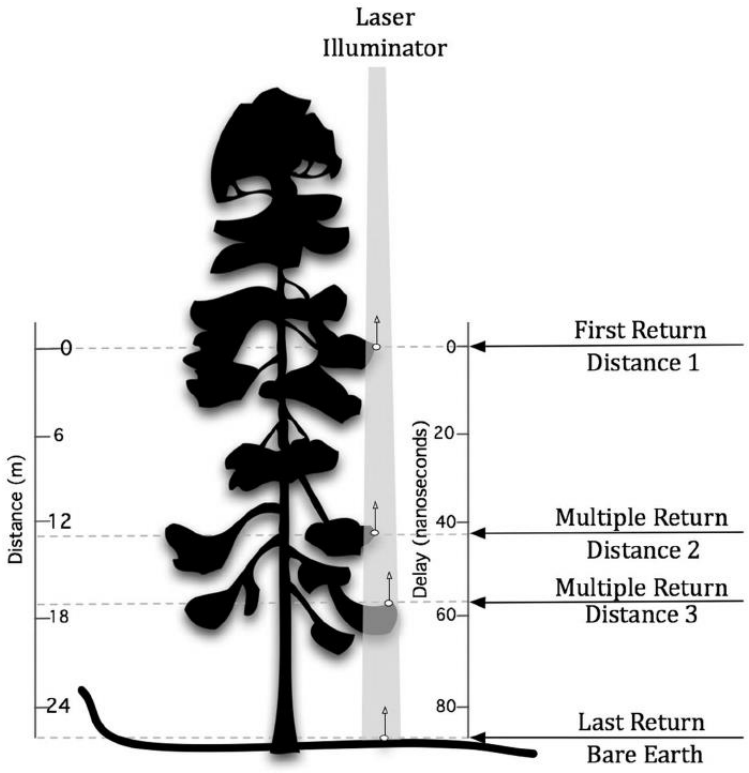
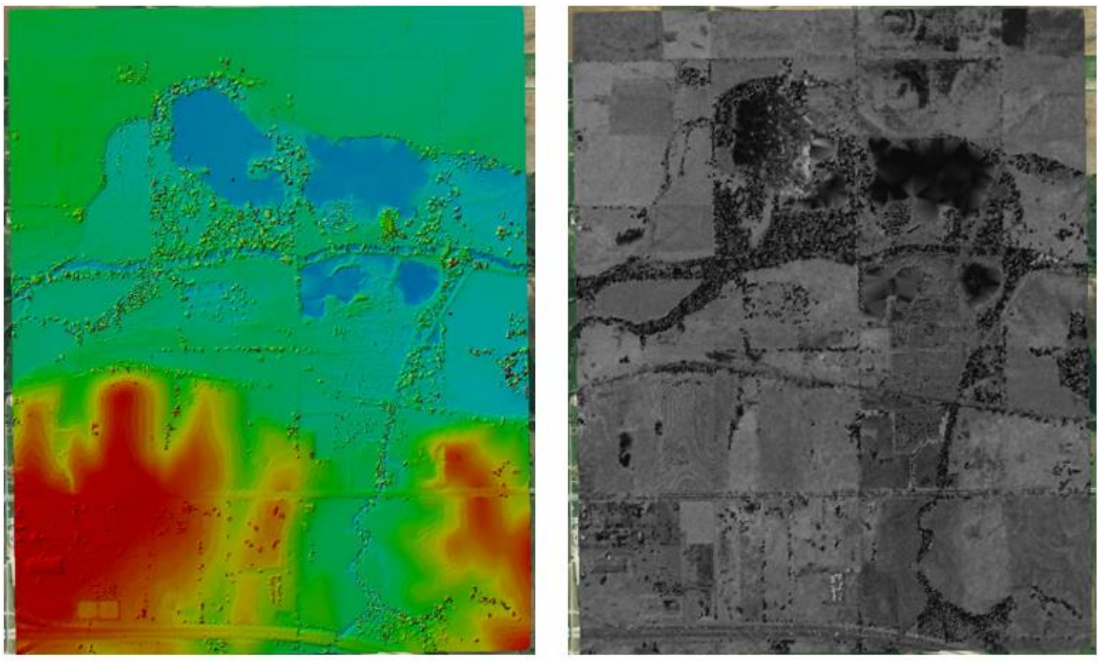


Figure 2 Multiple returns from a single laser pulse (Garcia et al., 2011)

The LiDAR acquisition system records, in addition to the laser returns, the strength of the returns (intensity). An example of the intensity map of a rural area is shown in Figure 3(b). LiDAR intensity is determined by the reflectivity of the surface object, so it can be used for land-cover classification.



(a) (b)

Figure 3 LiDAR point cloud (a) elevation and (b) intensity

There are different ways to represent the elevation map of a terrain: Digital Elevation Models (DEM), Digital Surface Models (DSM), Digital Terrain Models (DTM), and Triangular Irregular Networks (TIN). A DEM is a raster representation of the terrain's surface. A raster is a data structure which consists of a matrix of equally sized cells arranged in a grid where each cell contains an attribute value and location coordinates. DEM is interchangeably used with DSM and DTM. Digital surface model is a DEM created from the first return in a multiple-return LiDAR pulse or the sole return for a single-return pulse. Thus, DSM represents the earth surface and captures all natural and built features on it. Whereas, DTM is a DEM that includes only the bare earth surface. (See Figure 4).

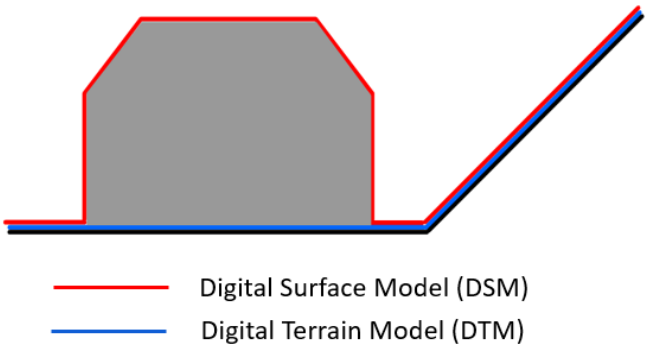


Figure 4 Surface represented by DSM and DTM

The most popular geometric property exploited by different feature extraction strategy from rasterized LiDAR data is the local elevation change. Natural features usually possess a low elevation slope compared to man-made features. Some natural features such as water surface and ground are usually approximately flat. In general, natural features, commonly referred to as landforms, are tangible landscape objects that have a characteristic shape and are bounded by topographic discontinuity (MacMillan, 2009). Man-made features introduce high elevation changes on the terrain surface. Thereby, they are easily distinguishable from the background natural fluctuation.

In recent years, a new kind of LiDAR system has emerged called the Geiger-mode sensor. The Geiger-mode LiDAR offers several advantages over existing linear mode LiDAR capability. This includes the capability to capture a denser point cloud over a large area more quickly and at a lower cost than the traditional linear scanners. Figure 5 shows an example of the Geiger-mode LiDAR

data of an urban area. This may necessitate changes in the way we extract buildings as discussed below.

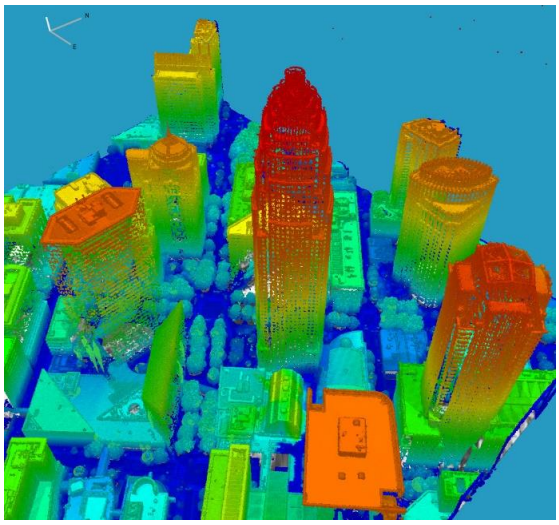


Figure 5 Geiger mode LiDAR data (Courtesy: Harris)

Man-made features such as buildings are usually extracted from LiDAR data based on the footprint outlines derived from roof edges, not the position of the wall (Awrangjeb, 2014). The reasoning behind that is aerial linear mode LiDAR has an almost nadir view, i.e., it captures the details of the rooftop while collecting only a few returns from the walls. A similar issue has to be dealt with for natural features such as a cliff or a deep ravine. Assuming the outline of the roof as the footprint of the building can sometimes be misleading, especially for the building, which has a larger rooftop than its footprint area (S. Wei, 2014).

With the advent of new LiDAR sensor technology such as Geiger-mode LiDAR and Single-photon LiDAR, the vertical walls or planes are no longer left out

from the laser illumination. Geiger-mode/Single-photon LiDAR's 360-degree (Figure 6) view guarantees that vertical features present in the man-made structure or the landform are fully covered. With a lot of points on the wall, it can now be mapped more accurately. Thus, the true footprint of the building and natural features with a steep slope such as cliff and deep ravine can be extracted.

(a) (b)

Figure 6 Laser footprint of (a) Linear and (b) Geiger-mode

The purpose of this dissertation, as stated in Section 1, is to introduce a robust new automated solution that extracts all the objects in a given LiDAR image with a high degree of accuracy. The method was developed with the mindset that it would exploit the data on the aspect of the object (vertical feature) recently made available by the new LiDAR sensor technology. Unlike the traditional segmentation methodology, the vertical features of the objects have been mapped in this method. However, the method is tested on traditional linear-mode LiDAR data and has demonstrated full competence. Despite low point density on the vertical plane,

the walls can still be constructed via Delaunay triangulation or Digital Elevation map generation, which connects the roof boundaries to the nearest points on the ground.

3. Review of the Literature

Single class object detection from airborne LiDAR data has been an important research topic for many years. In this dissertation, the term 'object' refers to its generalized form, including anthropogenic or man-made objects as well as natural objects. Extensive work has been performed on automatic detection of buildings (Verma et al.,2006; Sohn et al., 2008; Sun and Salvaggio,2013; Awrangjeb,2014), roads (Hu,2004; Hu-ying et al.,2012), trees (Koch et al.,2006; Liu et al.,2013), waterbody (Toscano et al.,2015; Acharjee et al.,2016) and vehicles (Yao et al.,2010). However, since in this work, we aim to extract a variety of object class present in the scene so we would focus on multi-class object extraction.

Multi-class object extraction from a given scene can be approached from two directions: simultaneous classification and hierarchical classification. Simultaneous classification strategy classifies each LiDAR point as belonging to a specific object in a single step. Whereas, hierarchical classification consists of more than one step. For example, a popular hierarchical classification technique involves two steps: filtering and object classification (Zhu and Toutin, 2013). In the filtration step, the point cloud is partitioned into two class: ground and non-ground. This is popularly known as a ground filter or DTM generator. Here, the overlying objects can be extracted indirectly by the difference between the DSM and the DTM generated. This representation of elevated objects is called normalized digital surface model (nDSM). The result can be fed into a multi-class classifier to identify

their classes. Most ground filter cover here have not been designed for the purpose of objects extraction, but they all can potentially be adapted to work in conjunction with the multi-class classifier for enabling object representation of the terrain.

Based on the basic element used in the classification process, multi-class object classification can be categorized into two classes: Pixel-based and Object-based. In the Pixel-based approach, each pixel of the image is classified based exclusively on the attribute of itself and its spatial neighborhood. Pixel based approach essentially employs the simultaneous classification strategy. In the Object based approach, the image is first partition into meaningful segments and then a classification is performed on those segments. Therefore, Object based approach adopts hierarchical classification strategy.

A typical LiDAR-based ground filter is a pixel-based approach, which classifies each pixel into one of two classes: ground or non-ground. The attributes are derived from the elevation value of the point.

3.1. Pixel-based object extraction

Extensive work has been reported in pixel-based object extraction. In the following, some of the popular Pixel-based object extraction methods are discussed:

- Antonarakis et al. (2008) proposed a supervised classification strategy exploiting the combined analysis of both skewness and kurtosis for both elevation and intensity distributions to distinguish between forest and ground.

- Lafarge and Mallet (2011) introduced a four-step urban objects extraction and modeling strategy from complex urban scenes. First, the point cloud is classified (unsupervised) into the building, vegetation, and ground using discriminative geometric features. The author combines local features and local context by applying an energy function and Potts model. In the second step, shapes and lines are extracted from the segments. In the third stage, the extracted components are projected in a 2D grid space arrangement by applying geometric constraints, and the labels evaluated in the first step are propagated accordingly. In the final stage, the urban objects are represented in 3D.

- Brodu and Lague (2012) classify complex natural scenes based on a multi-scale analysis of 3D points from terrestrial LiDAR. The analysis reveals, for each point, the optimal neighborhood size to be considered and the local 3D organization of the vicinity (linear, planar or volumetric). Such dimensional information at each point, and at different scales, are introduced as input features for classification. They demonstrate its efficacy by separating riparian vegetation from the ground and classifying a mountain stream into vegetation, rock, gravel, and water pixel.

- Niemeyer et al. (2013) proposed a supervised point-wise classification method that incorporates a random forest classifier into a Conditional Random Field (CRF) framework. The method successfully detected six types of urban

objects: natural ground, asphalt, buildings, vegetation, fences, and cars. However, a massive amount of training samples is required to establish a per-class and per-interclass relationship, which can be a logistic nightmare considering the huge variety of urban objects usually present in the urban scene.

- Weinmann et al. (2015) proposed a robust methodology that assigns each individual 3D points with a local 3D neighborhood of optimal size for extracting distinctive geometric features. The features are then fed along with training example to a supervised classifier.
- Guo et al. (2015) used JointBoost classifier to classify the point cloud into five main class: building, ground, vegetation, power-lines and pylons, using locally extracted features. The JointBoost classifier is known to perform automatic feature selection and therefore can process many input features for multi-class classification. As a post-processing step, a Graph-cut segmentation method is used to improve the classification results.

Existing ground filter methods can be classified into five major classes: surface model-based (Tóvári and Pfeifer, 2005; Mongus and Žalik, 2012; Chen et al., 2013; Chen et al., 2012; Chen et al., 2016), morphology-based (Arefi and Hahn, 2005), TIN-based (Axelsson, 2000), Segmentation-based (Chen et al.,2014; Zhang et el.,2013) and Statistical analysis based (Crosilla et al., 2013; Costantino et al.,2011). Here, some segmentation-based and morphological based (Arefi and Hahn, 2005) ground filter can be categorized as object-based, but the rest are

pixel-based. Here, we discuss some of these pixels based ground filter approaches, among which some were directly used for hierarchical classification:

- Axelsson (2000) proposed a ground filter algorithm based on iterative densification of the TIN representation of the terrain. The algorithm first identifies some control ground points and use them to create a sparse TIN model of the terrain. During each iteration, more and more points are added to the TIN if they meet certain criteria. As a result, the TIN model is progressively densified. The iteration continues until there is no point left that satisfy the criteria. A version of this algorithm has been implemented in a popular commercial software TerraScanTM from TerraSolid (Sithole and Vosselman, 2003). This method has proved very successful in urban areas. However, this filter performance is below par in case of rough terrain (Chen, 2016).

- Tovari and Pfeifer (2005) proposed a DTM generation method based on the surface Moving Least Square (MLS) linear interpolation, which has been highly popular among researchers. Firstly, the method iterates an interpolated surface to the ground using some control ground points. At each iteration, the residuals are calculated and a proportionate weight is assigned to each point. This minimizes the weights for points from non-ground objects whilst points from ground object have a large influence on the run of the surface. The iteration continues until the ground surface model stabilize.

- Meng et al. (2009) proposed a hierarchical classification strategy to extract building based on the morphological processing. The algorithm first employs a ground filtering algorithm to separate ground from its overlying objects. Next, an

analytic approach acts on the resulted nDSM to remove the remaining non-building points using building elementary structure filtering.

- Costantino et al. (2011) developed a classification strategy based on the statistical parameters of the LiDAR point cloud distribution values. In the unsupervised algorithm, a combined analysis of both skewness and kurtosis on elevation distribution classifies the point cloud distribution into terrain/off-terrain class. Next, the RANSAC algorithm has been applied on the off-terrain points to detect buildings. The heavy use of thresholds in the method potentially undermines the extent of automation involved in the process.

- Mongus and Zalik (2012) proposed a parameter-free strategy which combines surface-modeling, morphological filtering, multi-resolution comparison, and statistical analysis. The approach uses morphological filtering to eliminates noisy outliers, then use surface-interpolation at a multi-scale level to approximate the terrain. Next, the method employs a hierarchical morphological approach to compute the residual of each point from the surface. Finally, statistical analysis is applied enabling automatic thresholding and thereby, parameter-free ground filtering is achieved. The method is robust but computationally intensive.

- Chen et al. (2012) proposed a ground filtering method based on upward-fusion. This method first generates several preliminary DTMs at different scale level using a local minimum method. Next, by comparing the elevation difference between the course-scale preliminary DTMs and fine-scale DTMs with a threshold, the output DTM is refined. Unclassified points from the finer DTM that qualifies the criteria are selected, otherwise replaced by the value of the large-scale DTM. This

upward-fusion continues until all preliminary DTMs had been considered. The output is a refined DTM of high resolution.

3.2. Object-based object extraction

Can a computer be programmed to generate a hierarchical object representation from LiDAR data? The representation would present a hierarchy of super and sub-objects in mapping complex objects. There are very few examples of computer-aided hierarchical object representation. Object-based image analysis (OBIA) framework has gained traction recently in remote sensing and geographic information science (GIS) to map imagery into semantically meaningful objects. (Blaschke et al., 2014). OBIA involves two steps: image segmentation and object classification. Firstly, a segmentation algorithm is applied to partition the image into a relatively homogeneous group of pixels referred to as object candidates. This result, in addition to spectral values, in a plethora of new information such as aggregated spectral pixel values, morphology, texture, context as well as topology (Table 1). These features can be exploited in the subsequent identification of the object candidates. This makes object-based classification superior to pixel based classification. However, there is a caveat. Object-based classifier outperforms pixel based approach only if the resultant segments accurately represent real-world objects (Yan et al., 2008). In other words, the

segmentation quality has a direct impact on the classification accuracy (Duro et al., 2012).

A comparison of the features available for pixel and object-based classifiers is shown in Table 1.

Table 1 Features available for pixel and object-based method

	Pixel	Object
Spectral	✓	✓
Spectral statistics	×	✓
Size	×	✓
Shape	×	✓
Neighbors	×	✓
Hierarchy	×	✓

One other significant leverage the object based approach has over the pixel-based approach is that the image object delineation process reduces the effect of intra-class spectral variation and inter-class spectral similarity, thereby increase the classification accuracy (Wu, 2016).

One of the important aspects of the OBIA framework is the hierarchical image representation through a tree structure i.e. to produce a hierarchy of super- and sub-objects (Blaschke et al, 2014). As illustrated in Figure 7, hierarchical representation reveals two important pieces of topological information: the context

feature and the object's spatial arrangement feature. Availability of context information can lead to improvement in classification accuracy (Tang et al., 2013; Cui et al. 2016). It should be noted that the boundaries shared by the object and its sub-objects in the physical world, should perfectly overlap in their resultant segments of the image.

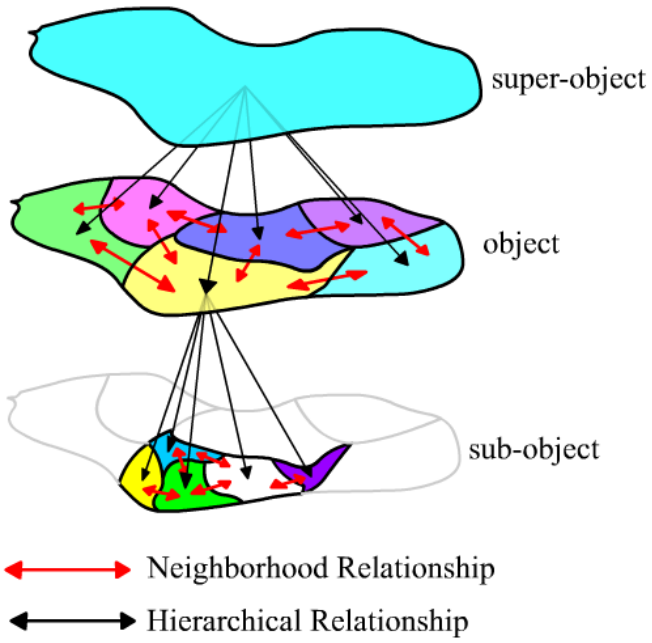


Figure 7 Hierarchy of image objects (Blaschke et al., 2014)

Relevant Segmentation techniques used in Object-based Image Analysis can be categorized into four classes, edge based, graph based, region based and ground filter based. In the edge based techniques, the LiDAR data is first converted to a raster and then the edges are detected. A contour generating algorithm is then used to close the region. The biggest limitation of the edge based technique is that

it cannot be used to extract objects which have overlapping surfaces such as bridges and buildings (Sithole, 2005). Among the four classes of object extraction method mentioned above, graph based technique, ground filter based, and the region based technique are the most popular.

In the graph based technique, the objects are first localized and then are extracted using graph cut algorithm. Some of the state of the art graph based object extraction techniques, used to interpret LiDAR data, are discussed below:

- Golovinskiy et al. (2009) present a min-cut based method for extracting small urban objects from a combined Airborne and Terrestrial LiDAR point cloud data. At first, a k-nearest neighbors graph is built. Then the potential object location is determined by applying a normalized cut upon the graph. Next, using points near those locations, foreground-background segmentation is computed with a min-cut algorithm that delineates the objects from its surrounding in the graph. Next, Shape features and contextual features are extracted for each point cluster. Finally, the feature vector for each candidate object is classified using a Support Vector Machine (SVM), trained on a set of manually labeled objects. A limitation of the segmentation method is that it often fails to detect all of the referenced objects.

- Yao et al. (2009) proposed a 3d segmentation method combining non-parametric clustering with spectral graph clustering to extract flyovers and vehicles in urban areas from raw airborne LiDAR data. The approach started with the mean-shift segmentation process, which over-segments the scene to produce superpixels. So the results are further classified via a modified normalized-cuts based on geometric features such as the horizontal and vertical similarity within

each superpixel, thereby generating a more consistent subset of laser points. The flyovers were extracted successfully, but the vehicles extraction rate was lower.

Region based techniques merge smaller regions into larger ones based on similarity. The growth stops once the growing region properties exceed the heterogeneity threshold. Among the region based techniques, the Multi-resolution segmentation (MRS) algorithm (Baatz and Schäpe, 2000) has recently garnered a lot of attention within the OBIA domain for the delineation of semantically meaningful objects. MRS is a bottom-up region growing method, which initially considers each pixel as a distinct object. Iteratively, smaller image-objects are merged into larger ones based on the similarity between adjacent image objects. The growth stops once the growing object properties exceed the heterogeneity threshold, defined by the scale parameter. Heterogeneity is usually expressed in terms of the spectral value and shape of the object.

Choosing the right scale parameter is crucial in this algorithm since it has a direct effect on the classification accuracy (Blaschke, 2010). The traditional procedure for scale parameter selection involves a long time-consuming trial-and-error. Even though this procedure allows the incorporation of expert knowledge, it is deemed irreproducible and not robust enough. Recently, some progress has been made in the development of methods for automatic tuning of scale parameters. For instance, the Estimation-of-Scale-Parameters (ESP) tool [Drăguţ, 2010] from eCognition is used by MRS to delineate landforms from the spectral image at multiple segmentation scales, whenever there is significant size variation. The resulting segments are then classified based on the physical attributes such

as length, width, and height and by the statistical frequency of specific geomorphometric variables [Evans, 2011].

- D'Oleire-Oltmanns, et al. (2012) employ multiresolution segmentation techniques on DEM to segment landforms such as drumlins since they vary significantly in form and shape. Unlike cirques and drumlins, most other landforms are not clearly bounded and their delimitation offers varying degrees of difficulty (Evans, 2012). For example, segmentation of gullies is difficult due to its heterogeneous morphologic characteristics; thus, the authors rely on aerial photographs for gully mapping. The paper also demonstrates the challenge facing MRS using ESP for delineating landforms such as drumlins. Despite the knowledge-based selection of an optimal segmentation layer, it performs poorly in delineating the exact limit of drumlins.

- Dragut, et al. [2013] employ MRS to segment elementary forms from Digital Elevation Maps (DEMs) based on the homogeneity of elevation derivatives such as gradient, aspect, profile curvature and plan curvature. However, landforms are composed of multiple such landform elements (Evans, 2012).

- Zhang et al. (2013) developed a land cover classification system based on object-based point cloud analysis. It consists of two sub-step: clustering and classification. For clustering, the author employs a plane-fitting algorithm to extract planar segments from the point cloud. Thirteen different features of the geometry, radiometry, topology, and LiDAR return characteristics are derived from the extracted segments. Next, a support vector machine (SVM) is used to classify the segments into five categories: ground, vegetation, building, vehicle, and powerline.

In the post processing, the author proposed a connected component analysis for 3D point clouds to improve the original classification results.

- Sevara et al. (2016) make a comparison of pixel-based and object-based approaches for the extraction of an archeological feature from LiDAR data.
- Wu et al. (2016) applied SVM classifier on combined MRS results from LiDAR nDSM image and WorldView-2 imagery to extract different urban land cover types.
- In another research effort (Chen and Gao, 2014), the land cover classification (building and trees) is based solely on the available features from discrete LiDAR data (elevation and intensity). Using just the following two key attributes: elevation and intensity difference between the first and the last return from LiDAR has proven to be effective in the object based classification. Here, eCognition has been employed for the multi-resolution segmentation.
- Xu et al. (2014) proposed a multiple-entity-based classification system where features were extracted from three different entities: points, plane segments, and segments produced by mean shift. Here, first, using the planar segments, the underlying ALS data is roughly classified to the ground, water, vegetation, roofs and undefined object. Then, from the labeled data, the walls and roof elements are identified point-wise using the contextual information of a building. Finally, from the points labeled as roof elements, the errors arise from contextual information are re-segmented by the mean shift method and then re-classified. The overall accuracy of the result is better than that achieved by using the just point-based method.

- Chen et al. (2016) proposed an image-segmentation-based method for urban DTM generation. Firstly, fine-scale segmentation is conducted on the DSM image using eCognition software. Next, among the unclassified segments, the lowest one within each cell is selected as the control seed ground segment. The rest of the unclassified segments are analyzed by comparing the spatial correlation with its nearest ground segment. In this process, all segments belong to ground are extracted. Finally, the output DTM is generated through post-interpolation.

, The ground filter based object extraction technique consists of two subsequent steps, filtering and object classification. In the filtration step, a pixel-based classifier such as a ground filter is employed to partition the point cloud into two class: ground and non-ground. The ground point is then used to create the DEM and the first return is used to create a raster DSM. Next, the overlying objects are extracted from the nDSM (which is the difference between the DSM and the DEM.). In the following, some of the well-known ground filter based object extraction techniques are discussed:

- Arefi et al., (2005) present a hierarchical segmentation procedure using morphological operations with different structuring element sizes to extract matched off-terrain regions. The region is then classified based on region properties. The authors targeted two classes: building and vegetation regions. The performance of the filter depends on the filtering window size. So, it needs an accurate a priori knowledge of the target terrain.

- Zhang et al. (2014) proposed an object-based point cloud analysis method to detect and extract vehicles from only one data source: LiDAR data. It involves two steps: candidate generation and verification. For candidate generation, firstly, a segmentation-based progressive TIN densification algorithm is employed to separate the terrains and off-terrains point and then, among the non-ground points, those point whose height within a scope is selected. For verification, firstly, features such as area, rectangularity, and elongateness are extracted, and then, rule-based classification is performed.

3.3. Challenges facing current object extraction methods

The challenges common to all object extraction approaches are as follows:

- Due to the relative complexity of the real-world objects, intra-class heterogeneity is high. This contributes to the over-segmentation problem (Sevara et al., 2016). Moreover, the problem is further exacerbated with the availability of higher resolution imagery, as the heterogeneity becomes more pronounced.
- Object extraction approaches focus on only protruded object such as building, trees etc. while ignoring intruded objects such as crater, crevice, swimming pool, irrigation channel etc.
- Selection of the appropriate scale parameters for segmenting out object and sub-objects to generate the hierarchical object representation.

The challenges facing MRS based region growing technique are as follows:

- The performance of MRS heavily depends on the scale parameter selected, which is challenging to obtain automatically for every object class present in the scene (Musci et al., 2013). This leads to over-segmentation and under-segmentation.
- Hierarchical relations among objects derived at different scales is difficult to establish. For example, the boundaries shared between objects and sub-objects in the real-world scenario may not coincide in their corresponding resultant segments due to the imprecise selection of the scale parameters.

The limitation of graph based object extraction techniques are as follows:

- The discriminative feature used in graph based object extraction methods is the proximity measure between neighbor nodes of the graph. In some objects, for example, a hill with a gradual slope, it is not clear where the hill ends and the flat ground begins. Graph-cut requires such clear-cut distinction. Therefore, it has been rarely used for extracting natural object
- Graph cuts algorithm is only capable of reaching a global optimum for two labels problems such as foreground/background image segmentation. Therefore, to accomplish hierarchical segmentation, a non-parametric clustering algorithm such as mean-shift is usually employed. However, the clustering algorithm is scale dependent. To illustrate the problem, an example of 2D mean-shift clustering of a feature vector of size 5 (color (R, G, B) + location (x, y)) with two different scale-

parameters (σ_s, σ_r) is shown in Figure 8. In Figure 8 (b), the Gaussian window is small so that small sub-objects such as windows, doors, pillars are segmented out. In Figure 8 (a), the Gaussian window is large, as a result, the intricate details are lost. However, the whole building (parent object) is segmented out. Both results are important as in one case the parent object is extracted whereas, in another case the, sub-objects are extracted. The challenge is, however, to find the appropriate scale-parameter that will serve the purpose.

$(a)\ \sigma_s = 50,\ \sigma_r = 5.0$ $(b)\ \sigma_s = 5,\ \sigma_r = 2.5$

σ_s: gaussian window applied to x, y parameters

σ_r: gaussian window applied to R, G, B parameters

Figure 8 Two mean-shift results with different parameters (Courtesy: IS-lab, Halmstad University)

- Objects that lie in close proximity are difficult to separate using the graph-cut algorithm. This is again because the algorithm relies on proximity between neighbor nodes for discrimination.

The challenges facing a ground filter based object extraction technique are as follows:

- These object extraction techniques employ a ground filter in the first step. Therefore, the quality of the filtration has a direct impact on its performance. The inaccuracy in the filter results is defined by two statistical measure: Type I error and Type II error. A Type I error is the incorrect rejection of bare earth points whereas, a type II error is the failure to reject object points. There is a trade-off between type I error and type II error. Most filters are designed to minimize Type II errors. The filter parameter is set such that it guarantees the removal of most object points, even those objects that have a small size and a relatively low height. However, this may cause the removal of many ground points and thereby increases the Type I error.

- Most ground filtering methods mentioned above have one thing in common; they are all pixel-based approach. They classify each LiDAR point as belonging to an object or ground based on its attribute and its neighborhood. Consequently, the classifier in the filter algorithm has limited information to work with and is deprived of the contextual information.

- The result from pixel-based approach such as ground filter may contain some misclassified isolated pixels or group of pixels, which collectively known as salt and pepper noise.

- In the case of ground filter object extraction approach, as far as I know, there is no solution available to produce hierarchical object representation.

3.4. Conclusion

Based on the literature review, the following conclusions are drawn:

- Object based approach is superior to pixel based approach. Therefore, in our research, we focused on developing an object based approach to extract 3D objects.
- In an ideal situation, the resulting segment presented to the object based classifier would correspond to the real-world object. However, current object extraction techniques fall short of the mark. They are, therefore, cannot fully exploit the leverage the object based approach has over the pixel-based approach. We attempt to overcome these limitations in our proposed methodology.
- The main two challenges facing OBIA are scale dependency and the relative complexity of the real-world objects.

4. AVSOE Algorithm Ideation

The aim of the research has been the development and the implementation of an algorithm for automated extraction of unknown salient objects - natural and man-made, protruded and intruded, from the topographic surface using airborne LiDAR data. As discussed in the previous section, the main two challenges facing object based image analysis are the scale dependency and the relative complexity of the real-world objects. Our object extraction algorithm is designed to tackle these problems. In this section, the underlying assumptions upon which our object extraction algorithm is based, are first discussed. Thereafter, a brief overview is provided on the working principle of the algorithm.

4.1. Characteristics of an object

Axiom 1: An object can be defined as one which introduces a distinct geometric concavity or convexity on its topographic background.

The following two corollaries can be derived from Axiom 1 (referring to Figure 9 and Figure 10):

Corollary 1: An object is bounded by topographic discontinuities.

Corollary 2: An object can be approximated into either of (or a mixture of) two fundamental 3D geometric structures: a '*convex-like'* object and a '*concave-like'* object.

Definition 1: A convex-like object is one which introduce convexity to the topographic background

Definition 2: A concave_like object is one which introduce concavity to the topographic background

For example, buildings, drumlins, hills, trees etc. falls into the convex-like structure category whereas a watershed, a swimming pool, a gully, a crater etc. can be categorized as concave-like structures. The volcano has characteristics of both types.

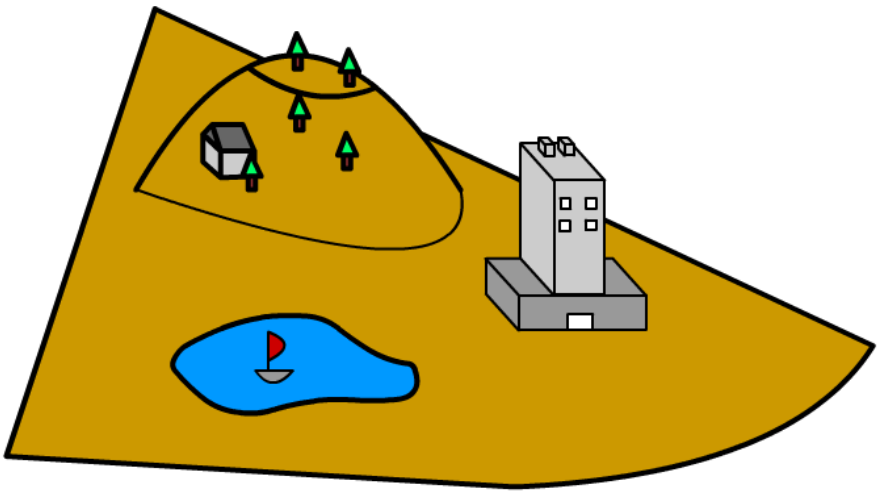


Figure 9 Objects and its topographic background

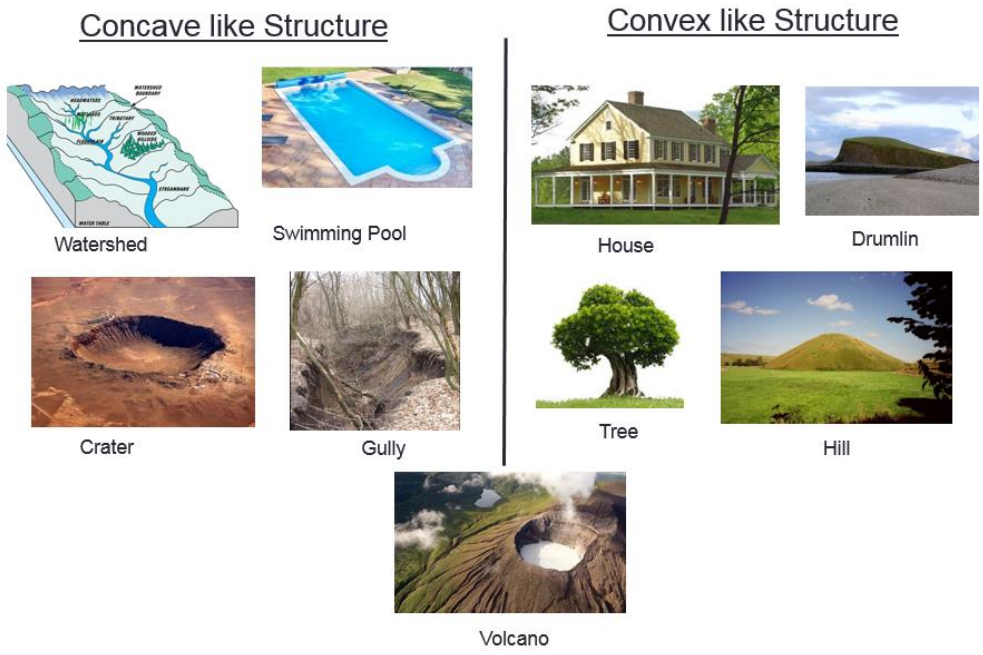


Figure 10 Real-world objects types

We can decompose any concave or convex-like object into two surface types: '*slope surface*' and '*horizontal flat surface*'.

Definition 3: A slope surface is one which encompasses the object or sub-object and whose slope is strictly monotonic.

Definition 4: The horizontal flat surface is an open surface whose slope is zero.

An example is shown in Figure 11. Here, the surface of each object has been decomposed into multiple slope surfaces and horizontal flat surfaces.

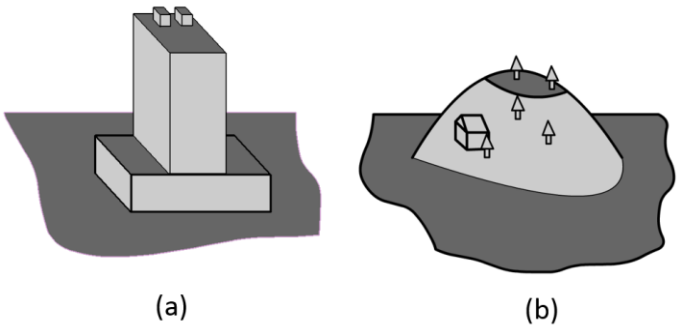


Figure 11 Decomposing a (a) complex building structure, and a (b) mountain into slope surfaces (lighter shade) and horizontal flat surfaces (darker shade)

Referring to Figure 11(a), a single slope surface would include all the vertical facades of the tall building. A second single slope surface would include all four facades of the shorter building. In Figure 11(b), a single slope surface would include the lighter shaded surface going all around the mountain. Virtual Surveyor based segmentation algorithm is designed to map only the slope surface of the object leaving behind the horizontal flat surface.

Axiom 2: A single slope surface always connects the object, no matter how complex, to its background. This special slope surface is called '*foothill_slope'*. Foothill_slope of an object ends with the boundary of the object.

Figure 12 shows the foothill_slope of the objects as illustrated in Figure 11. Referring to this figure and axiom 2, the following conclusion can be drawn,

Corollary 3: A seed growing segmentation algorithm that maps the slope surface of an object and which starts from a seed positioned in the vicinity of the

object boundary is guaranteed to capture the object boundary in the resultant segment.

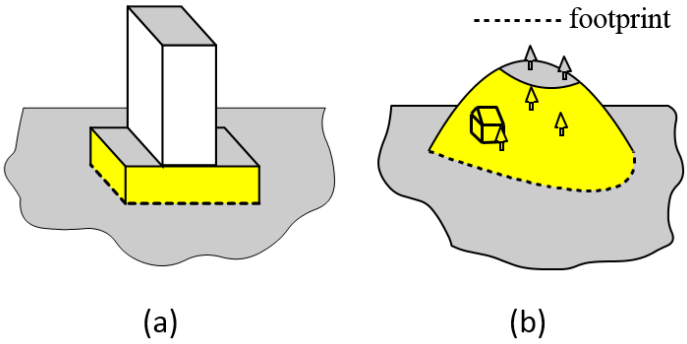


Figure 12 Foothill_slope (yellow shade) and footprint (dash line) of a (a) complex building structure, and (a) mountain

Our strategy is, therefore, to map the foothill_slope of the object using a seed growing segmentation algorithm. It starts with a seed at somewhere on the foothill_slope of the object and it includes neighbor that maintain the strict monotonicity as that defined by the slope of the seed. (See Definition 1).

In the process of mapping, the foothill_slope of sub-objects that lies on the foothill_slope region of the parent object are also get mapped and included into the growing segment. This is demonstrated in Figure 13 which shows the close-up view of the house on the mountain (Figure 11(b)). While mapping the foothill_slope of the mountain, the foothill_slope of the house is also get mapped. This is

because, surface B of the house maintain the strict monotonicity as that defined by the neighbor surface A of the mountain.

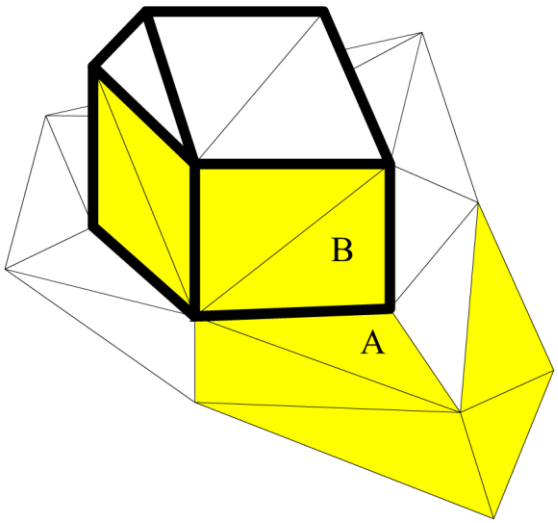


Figure 13 While mapping the foothill_slope of the mountain, the foothill_slope of the house is gets mapped.

A foothill_slope can be classified into two types: convex and concave.

Observation 1: Surface Normal vectors to all points in the foothill_slope of convex type point outward as shown in Figure 14.

Observation 2: Surface Normal vectors to all points in the foothill_slope of concave type point outward

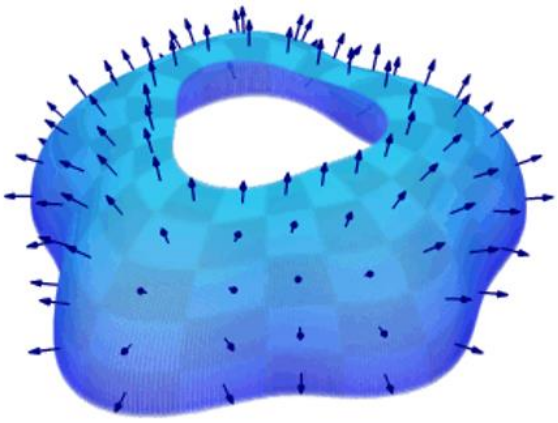


Figure 14 Normal to all points in the foothill_slope of a convex object point outward [Courtesy, 1ucasvb]

Referring to definition e and Figure 10, the following observation of the foothill_slope can be made:

Axiom 3: Foothill_slope of a convex-like and concave-like object is always convex and concave respectively. This means that a convex object can have several instances of horizontal flat surfaces, and/or concave slope surfaces, but its foothill_slope is always convex.

Definition 5: There exist a set of circular loops, associated to its foothill_slope, that encircle the entire structure and the normal to each point of the loop, points inward in the case of concave foothill_slope and outward in case of convex foothill_slope. For each point on the slope surface, there exists at least one member of that set of loops, which passes through the point. These special circular loops are called '*path_loops'*.

Figure 15 shows few path_loops encircling a mountain and a cavity. It is to be noted that the boundary of the object is itself a path_loop.

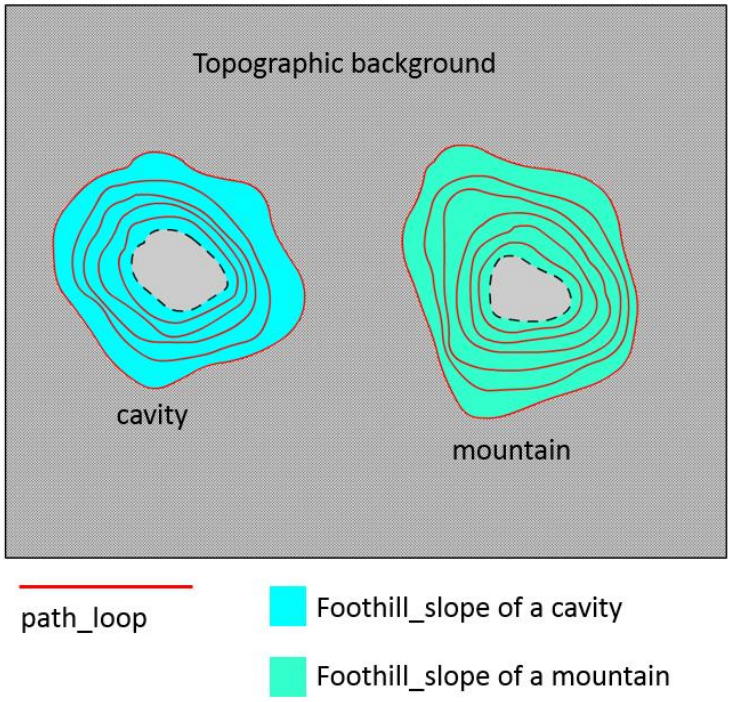


Figure 15 Top view of a cavity, a mountain, and its topographic background. Their foothill_slope and path_loops are also shown

According to the definition 5, observation 1 and 2, and axiom 3, the following conclusion can be drawn:

Corollary 4: the normal at each point of the path_loop either point inward or outward in case of concave or convex-like object respectively.

The following conclusion can be derived from axiom 2, definition 5 and corollary 4:

Corollary 5: Each object in a topographic map can be represented by a set of path_loops encircling the object. The normal at each point of the circular loop point inward or outward, depending on the type of the object.

The direction of the path_loops can be decomposed into two major components: cross slope and main slope. Here cross slope is perpendicular to the direction of the main slope as shown in Figure 16.

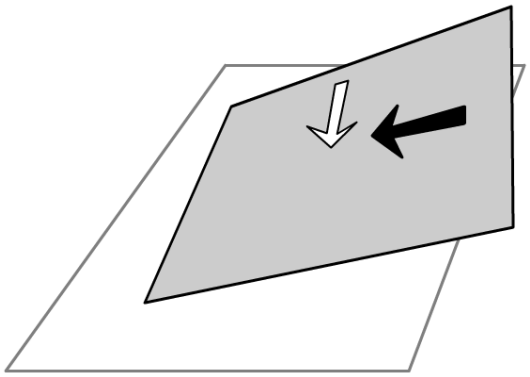


Figure 16 The black thick arrow indicates the direction of the main slope; the white arrow indicates the direction of the cross slope

An important observation of these path_loops is:

Axiom 4: The cross slope component of the path_loop dominates over the main slope component. This is demonstrated in Figure 17.

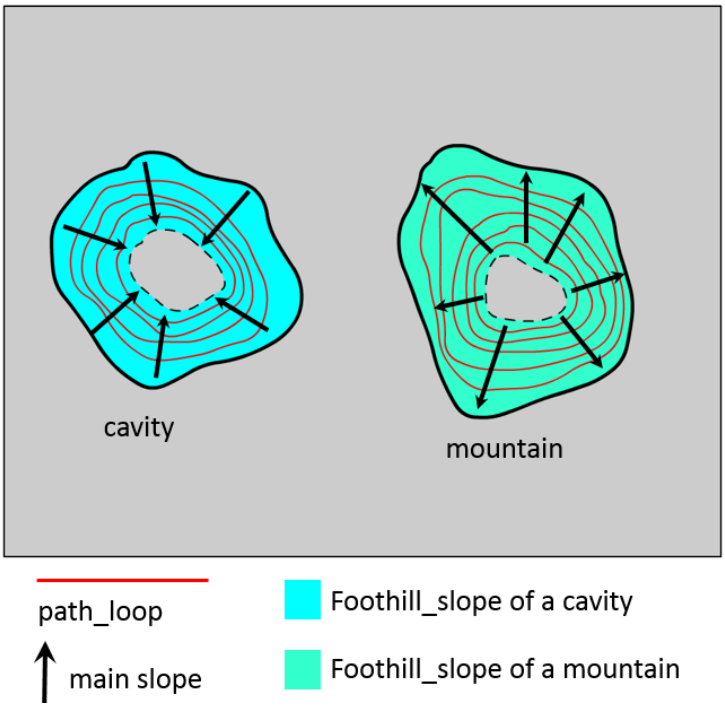


Figure 17 Cross slope component of the path_loop dominates over the main slope component

Like every other algorithm that deals with airborne LiDAR point cloud data, our algorithm first derives the surface representation from the point cloud model of the terrain. In our case, a network of the planar segment is constructed from the point cloud model of the terrain. This is done by fitting planes onto the point cloud. Figure 18 shows the stylistic simulation of the partition of a terrain (Figure 15) into a network of planar segments. The following conclusion can be derived from corollary 1:

Corollary 6: the planar segments border coincides with the object border as shown in Figure 18.

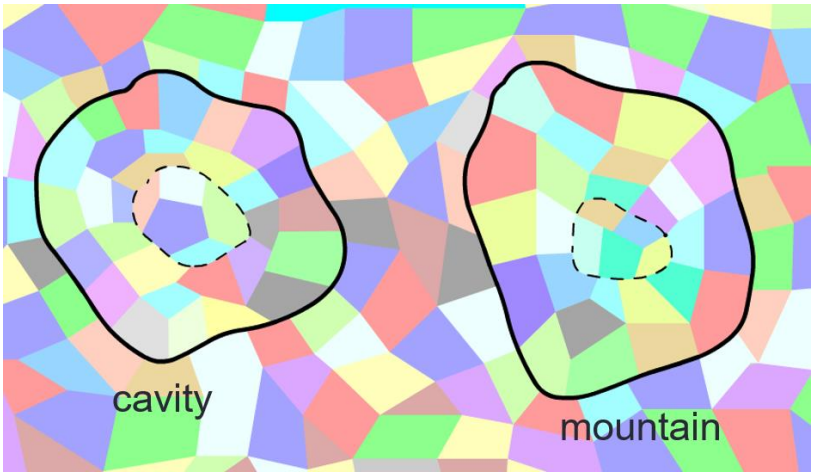


Figure 18 Partitioning the terrain (shown in Figure 15) into a network of planar segments. Here, each individual color represents a planar segment.

4.2. Surveyor Guidance Vector, on_slope condition and path_loop condition

Consider a virtual surveyor, as shown in Figure 19(b), standing on the sloping surface of a bowl-shaped structure, as shown in Figure 19(a). Now suppose, the normal to the background onto which the bowl is placed is known and is referred as Background Normal vector (BN vector). Here, the BN vector is vertical. The normal to the slope on which the surveyor is standing can be locally estimated and is referred as a Surface Normal vector (SN vector). The cross

product between the BN vector and SN vector produce a special vector which we called Surveyor Guidance vector (SG vector).

$$\vec{r} = \hat{z} \times \hat{n} = |\hat{z}||\hat{n}|sin\theta\hat{r} = sin\theta\hat{r} \qquad \textit{eq. 1}$$

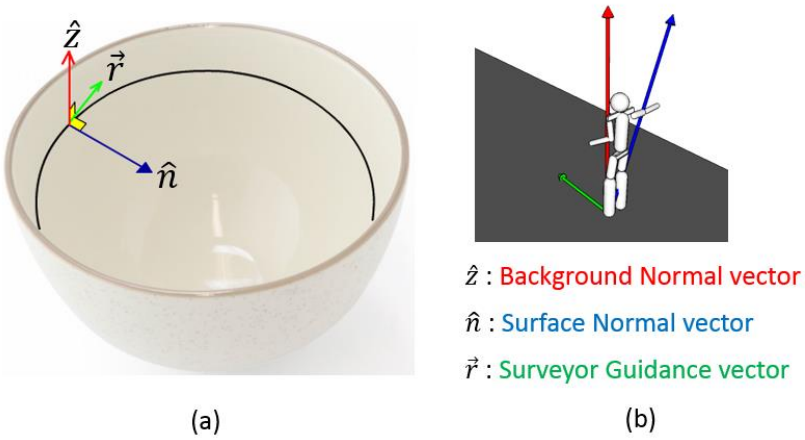


Figure 19 A surveyor as shown in (b), standing on the slope of a bowl structure as shown in (a)

Observation 3: If the surveyor is facing the direction of the Surveyor Guidance vector $\vec{r}$, the Surface Normal vector $\hat{n}$, the surveyor and his right arm are in the same plane. The following conclusion can be drawn:

Corollary 7: When the dot product between path vector $\vec{v}$ and the Surveyor Guidance vector $\vec{r}$ is positive, the Surface Normal vector $\hat{n}$ lies on her right-hand side. If the dot product is negative, SN vector $\hat{n}$ is on her left-hand side. The premise is proved below.

Proof: Figure 20 shows the top view of the surveyor standing on the slope. Here, the BN vector points out of the page. The path vector $\vec{v}$ shows the direction taken by the surveyor. The dot product between $\vec{v}$ and $\vec{r}$ is the scalar projection of $\vec{v}$ to $\vec{r}$ which is represented by the cyan colored vector in Figure 20.

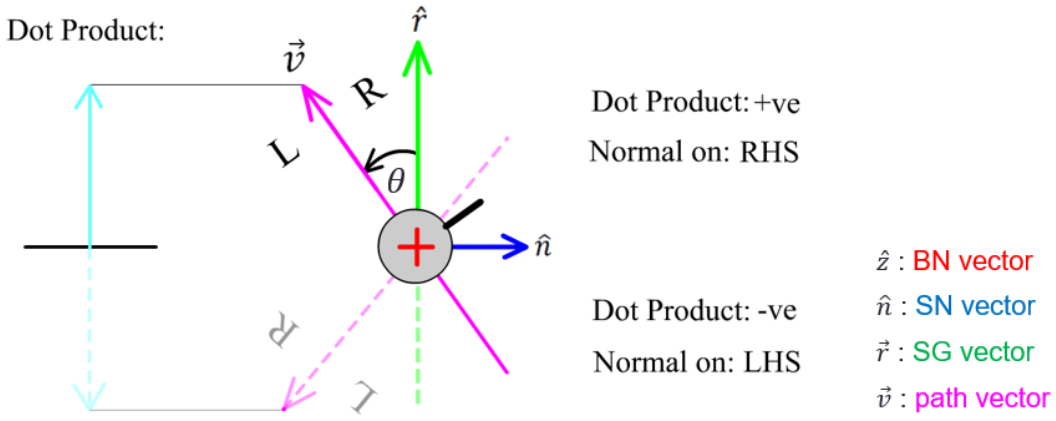


Figure 20 Surveyor path vector relation with SG vector

The dot product between $\vec{v}$ and $\vec{r}$ is given by

$$\vec{v}.\vec{r} = |\vec{v}||\vec{r}|\cos\theta$$

Now, when $\theta = -\frac{\pi}{2} \rightarrow \frac{\pi}{2}$, $\vec{v}.\vec{r} \geq 0$, SN vector is on RHS

And, when $\theta = \frac{\pi}{2} \rightarrow \frac{3\pi}{2}$, $\vec{v}.\vec{r} < 0$, SN vector is on LHS

Definition 5: Suppose a surveyor follows the arc shown in Figure 21. The path vector is shown in three different location of the arc. The path vector is tangential at each point of the curvy path. The arc can be divided into infinite pieces called '*path_elements'*. Each infinitesimal piece can be approximated as a straight

line which lies in the direction of the path vector at the start point. That means, at each point of the path_element, the path vectors point in the same direction along the path_element. Representing the path taken by a surveyor as a chain of path_elements concur with our surface representation of a network of planar segments. Here, the surveyor can hop from one planar segment to the neighbor planar segment in a straight line. Each such straight line represents a path_element.

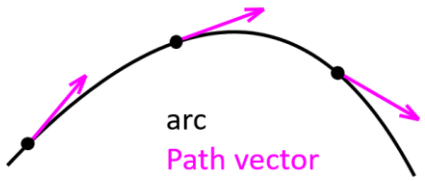


Figure 21 Path vector shown in three locations of the arc

Definition 6: Figure 22 shows an arc (arc_{AB}) drawn on the foothill_slope of a bowl-shaped structure. The direction of the arc is radially outward. The arc can be divided into its path_elements. At each point of each path_element, the dot product between the path vector and the Surveyor Guidance vector at that location is zero since they are perpendicular to each other. We call these special path_elements as '*radial_path_elements'*. Therefore, it can be stated that, at each point of the foothill_slope, a path_element can be associated such that the dot

product between the SG vector and the path vector is zero at all points of the path_element.

Figure 22 Occasion when the dot product is zero between the path vector and the SG vector

Corollary 8: At each point of the foothill_slope, the set of all possible path_elements, which can be associated to the point, excluding the radial_path_elements obeys the following condition. The dot product between the path vector and the Surveyor Guidance vector maintain the same sign at all points of the path_element. This premise is proved as follows.

Proof: In Figure 23, consider the set of points in the circle drawn on the foothill_slope of the concave-like structure. Consider the set of arcs where each starts from any of those points and ends at the center. Each arc can be divided into its path_elements. Each associated path vector of a path_element can be decomposed into two components: one in the direction of radial_path_element ($\vec{v}_r$)

and the other one is perpendicular to it ($\vec{v}_p$). Since, the radial_path_element component ($\vec{v}_r$) has no contribution to the dot product, as discussed in definition 6, the evaluation of the dot product is therefore solely depended on the component that is perpendicular to the radial_path_element ($\vec{v}_p$). Since the Background Normal vector is vertical, $\vec{v}_p$ can be visualize from the top view as shown in Figure 23. Figure 23 demonstrates that, at each point of $\vec{v}_p$, for each path_element, the normal points to the same side, either left-hand side or right-hand side, of the path vector component $\vec{v}_p$. That means, the dot product between the path vector and the Surveyor Guidance vector maintain the same sign at all points constituting the path_elements that belongs to the foothill_slope.

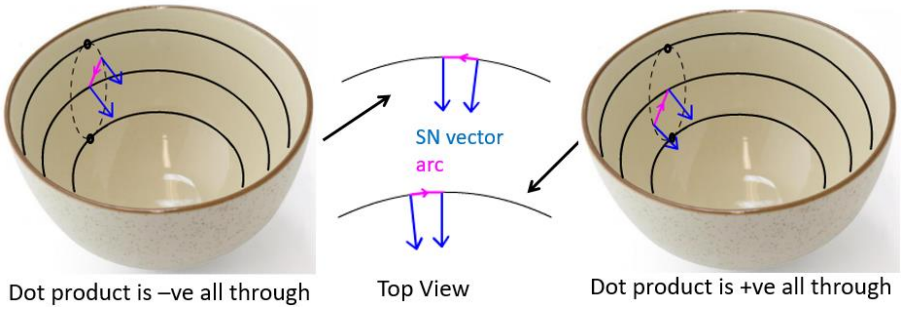


Figure 23 Occasion when the dot product is not zero between the path vector and the SG vector

Conclusion 1: If Antecedent S is a “slope surface” and Consequence N represent the following event, “dot product between the SG vector and the path

vector defined by two infinitely close neighbor points (except those pair of points which constitute the radial_arc) has the same sign". Then, S guarantees N, i.e. S => N. (Sufficiency condition).

Corollary 9: In the case of two close neighboring points, where each belongs to a different slope surface, the dot product at all points of the path_element connecting the points do not have the same sign. This premise is proved as follows.

Proof: In Figure 24 (a), plane A and B belong to the same slope surface since the monotonicity of their grade has been maintained, whereas, in (b) and (c), they each belong to a different slope surface. In Figure 24 (a), $\vec{n}_A$ and $\vec{n}_B$ both point to the same side of path vector of the path_element defined by any pair of points (except radial_path_element) that lies on both side of the boundary between A and B. On the other hand, in Figure 24(b) $\vec{n}_B = 0$ and in Figure 24 (c) $\vec{n}_A$ and $\vec{n}_B$ both point to the opposite sides of the path vector which connects the two neighboring points across the boundary. So, the dot product at either end has the same sign in (a) but not in the case of (b) and (c).

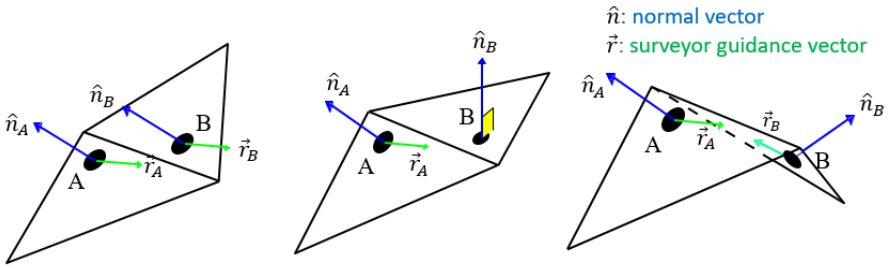


Figure 24 Adjacent planes when (a) form a slope, (b) and (c) does not form a slope.

Conclusion 2: Again, if the Antecedent S is “slope surface” and the Consequence N represent the following event, ““dot product between the SG vector and the path vector defined by two infinitely close neighbor points (except those pair of points which constitute the radial_arc) has the same sign” then the following condition statement holds: if S is false, N is false or in other words, S <= N (Necessity condition)

Conclusion 1 and 2 can be summarized as S is necessary and sufficient for N or in other words, S if and only if N i.e., S ⇔ N. That means, the adjacent planes are part of the same surface slope if and only if the dot product between the path vector $\vec{v}$ and the Surveyor Guidance vector $\vec{r}$ have the same sign all through.

Referring to Figure 25, according to the conclusion derived above, the point A and the point B lies on the same slope surface if,

$$\vec{v}_{AB}.\ \vec{r}_A \neq 0 \qquad \text{eq. 2}$$

$$\vec{v}_{AB}.\ \vec{r}_B \neq 0 \qquad \text{eq. 3}$$

$$sign(\vec{v}_{AB}.\ \vec{r}_A) = sign(\vec{v}_{AB}.\ \vec{r}_B) \quad \text{eq. 4}$$

This expression (eq. 4) can be rewritten as follows:

$$(first\_dot\_prod) \times (second\_dot\_prod) > 0 \quad \text{eq. 5}$$

Where, $first\_dot\_prod \ = \vec{v}_{AB}.\ \vec{r}_A$ and $second\_dot\_prod \ = \vec{v}_{AB}.\ \vec{r}_B$

This expression (eq. 5) is termed as '*on_slope*' condition. Therefore, the planar segments that satisfy the above condition belong to the same slope surface.

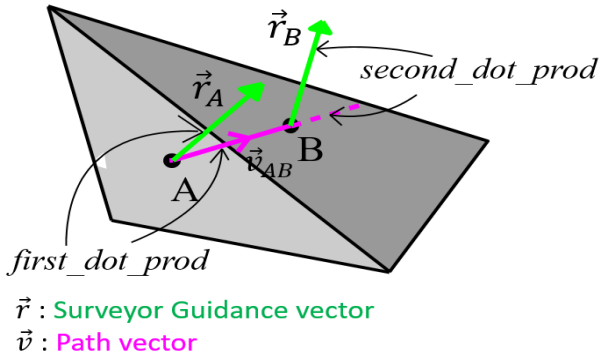


Figure 25 Transverse across adjacent planar segments

The above expression can be used for seed growing object segmentation process. The planar segments that satisfy the condition are only eligible for inclusion in the object image segment.

It is to be noted that in our surface representation in the form of planar segments network, the resolution of the network needs to be very high for the condition required by premise 2 and 3 to be effective. However, to generate such

high-resolution network of planar segments is often not possible. Therefore, even though theoretically, the implement of the on_slope condition in the segmentation process is sufficient to guarantee the accurate mapping of foothill_slope of the object. However, practically on many occasion, due to the granular representation of the terrain, the on_slope condition fails to serve this purpose.

An example of such case is shown in Figure 26. The highest resolution possible for our network of planar segments is limited by the raster representation of the point cloud from which the network has been derived. Details of the construction of the network of planar segments are given in section 5.1.

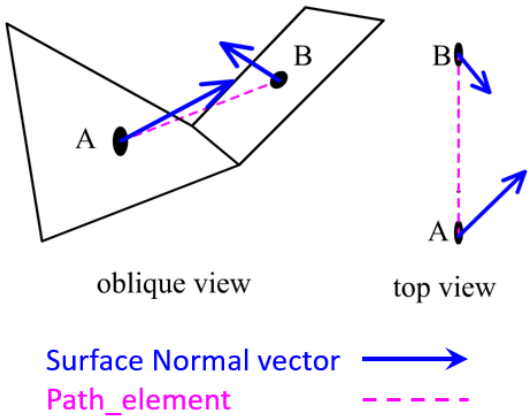


Figure 26 Case when on_slope condition fails

Here, as shown in the figure above, the elementary unit of the raster such as A can have such a distorted nonplanar geometric surface. As a result, the Surface Normal vector of both cell A and B point to the RHS of the surveyor, so in both cases, the dot product has the same sign. These kinds of cases render the

necessity condition of the on_slope condition (conclusion 2) ineffective. That is why the application of the on_slope condition for mapping the foothill_slope is not enough and so it is just a part of the elaborate process described in chapter 7 and 8.

Referring to Figure 27, suppose a virtual surveyor diligently follows one of the path_loop shown in Figure 15. Since, at each point of the path, the SN vector points to the right-hand side of the surveyor so, the dot product between the path vector and SG vector is positive all throughout the path_loop.

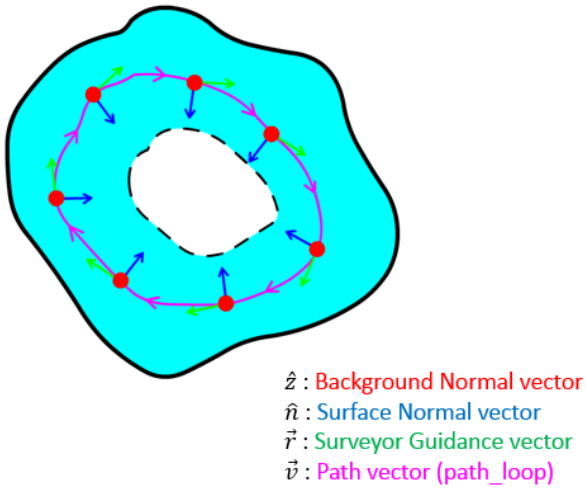


Figure 27 Virtual surveyor following a path_loop

Therefore, referring to Figure 25, suppose planar segments A and B are neighbors at somewhere on the foothill_slope, then $\vec{v}_{AB}$ constitute a path_loop if the following condition hold:

$$(first\_dot\_prod > 0) \wedge (second\_dot\_prod > 0) \qquad \text{eq. 6}$$

Eq. 6 is a special case of eq. 5. This is a property of the path_loop and is therefore called ‘path_loop condition’.

Now, this property of a path_loop has three major implications:

1. If there exists an arbitrary loop on a topographic surface, and if it is found that the loop obeys the path_loop condition, then the loop is a path_loop. That means, there is an object present and the loop encircles it.
2. Since the dot product is positive all through the path_loop. That means, the SG vector can be used to guide the surveyor around the foothill_slope.
3. The direction of the path_loop for a concave foothill_slope is clockwise whereas, the direction of the path_loop for a convex foothill_slope is counter-clockwise.

4.3. Overview of the AVSOE algorithm

Before delving into the details of the proposed methodology, this section walks through the logical steps that have been taken which eventually leads to the final form of the algorithm.

First, a network of the planar segment is constructed from the point cloud model of the terrain as discussed in section 4.1. Next, a directed graph is constructed from the planar segment representation of the terrain, with each planar segment represent the node and there exist an edge between neighboring nodes only if the edge complies with the *path_loop* condition. According to the path_loop

property discussed in section 4.2, it is guaranteed that this process will produce a set of a chain of connecting edges in the graph that encircles the foothill_slope of each object and sub-object present in the scene. Thereby, these cycle graphs (circular loops) represent the path_loops. Detection of these path_loops will reveal simultaneously the presence and position of all objects and sub-objects in the scene. However, such graph can be very dense for a large terrain, so finding each loop in the graph can be computationally expensive. Moreover, the ground itself has plenty of small concave and convex structures. So, loops will form around them and hence contribute to the set of false positive. To overcome these problems, a best-of-class pixel based ground filter algorithm can be employed to filter out most of the ground planar segments. Analyzing the filter result provides crucial information about the object candidates, such as seeds and back ground normal vector, which can be utilized by a seed growing segmentation algorithm to carve out regions from the map, where each region includes the foothill_slope of an object candidate. It is to be noted that the object candidate is a sub-set of the segmented region. These segments are called ‘naïve segments’ and the process that produces them are called ‘naïve object segmentation’. After the segmentation, a sub-algorithm is assigned to work on the sub-graph contained inside each of the regions individually to extract the boundary of the object and its sub-objects. The result is a hierarchical representation of the object and the process that produce it is called hierarchical decomposition algorithm. This divide and conquer approach of AVSOE made the overall process very efficient and remove most of the false positives.

Figure 28 shows the overall block diagram of the Automatic Virtual Surveyor based Object Extraction (AVSOE) algorithm. It consists of three stages: Preprocessing, Naïve Object Segmentation and Hierarchical Decomposition. The preprocessing step employs the ground filter and extracts the object candidates, each of which is represented by its corresponding seed and Background Normal vector. The result is fed into the Naïve Segmentation step which produces the naïve segments out of it. The Hierarchical Decomposition module works on the sub-graph contained inside each of the naïve segment and extract the object and its sub-objects boundaries.

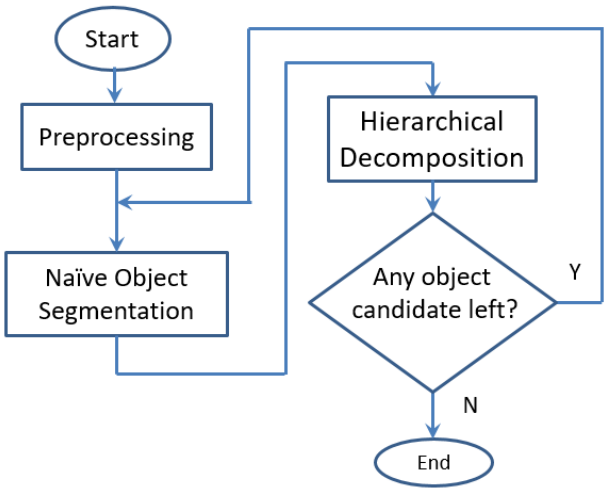


Figure 28 Flowchart of the proposed methodology (AVSOE)

Figure 29 shows the stylistically simulated results obtained from each of the three stages. In the following chapters, the individual parts of the flowchart are explained in detail.

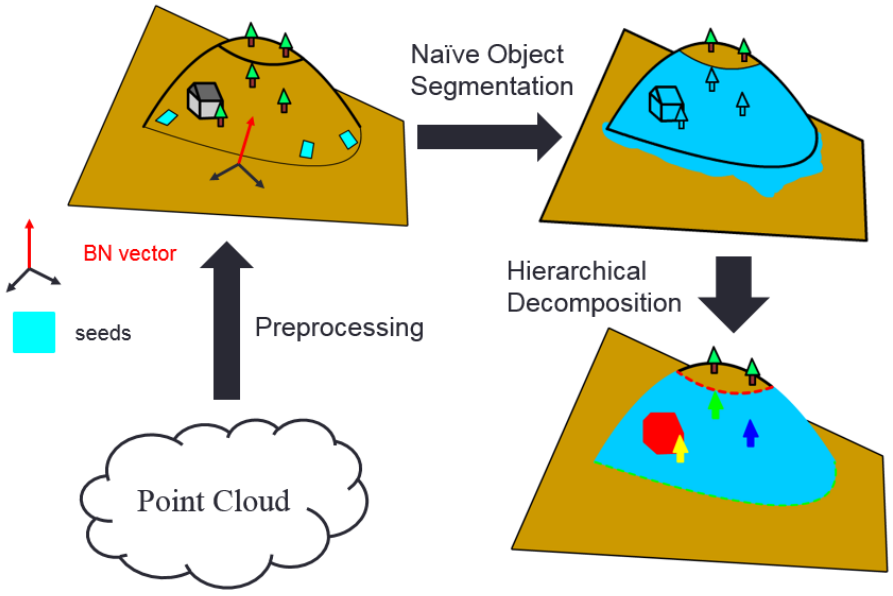


Figure 29 Stylistically simulated result obtained from each of the three main stages of the Virtual Surveyor object extraction (VSOE) algorithm

5. Preprocessing

In this section, the preprocessing stage is described in detail. This sequence of steps processes the input point cloud data and extract information that is later used by the subsequent stages.

Figure 30 shows the overall block diagram of the Automatic Virtual Surveyor based Object Extraction (AVSOE) algorithm with preprocessing stage in details.

1. The input to our algorithm is the LiDAR point cloud data of the target terrain.
2. In the preprocessing step, the point cloud is first rasterized to produce a Digital Elevation Model (DEM) representation.
3. The mesh is then fed into a plane-fitting algorithm to generate planes.
4. Our method then employs a well-known pixel-based ground filter (Tóvári and Pfeifer, 2005), whose sole purpose is to filter out the background planar segments (ground pixels) and present the salient planar segments (object pixels) to the subsequent object extraction algorithm.
5. Since neighboring object planar segments may have derived from a single object, the planar segments are grouped together using connected component analysis. Each group of planar segments provides a number of seed candidates which is leveraged by our (seed-dependent) naïve object extraction sub-algorithm. Background Normal vector is also extracted by fitting planes to the neighboring planar segments that belong to the ground.

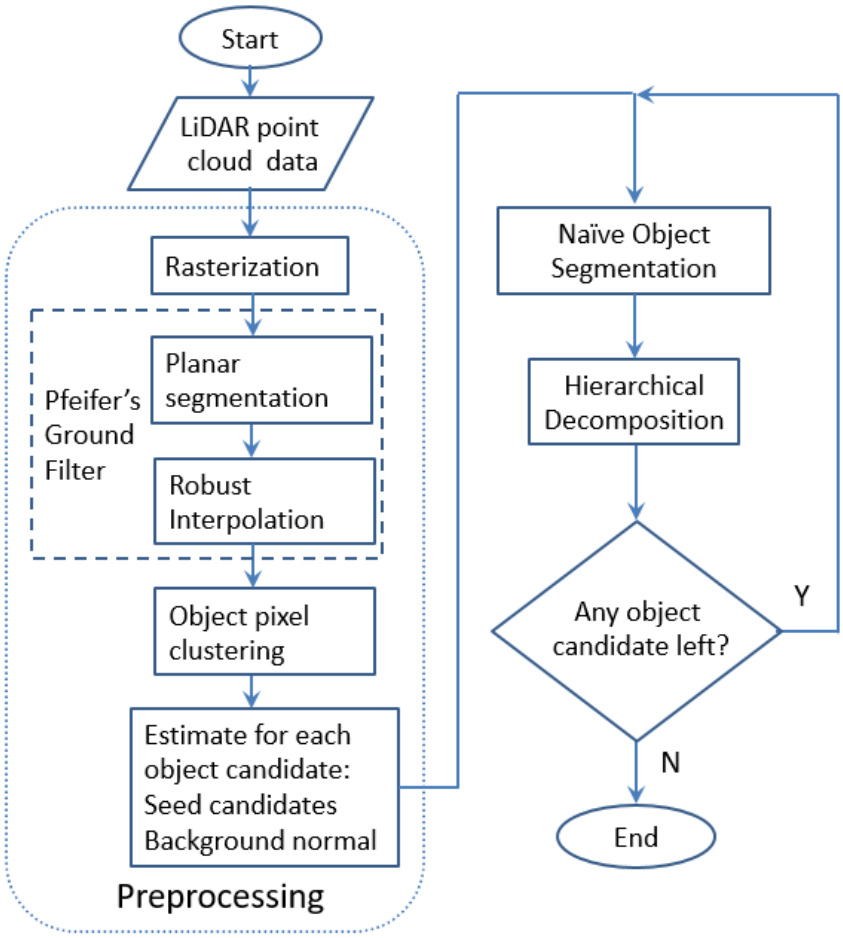


Figure 30 Flowchart of the proposed methodology (AVSOE)

5.1. Rasterization

As discussed in Chapter 2, the two key derivatives that can be produced from the LiDAR point cloud data for the surface analysis are Digital Elevation Models (DEMs) and Triangulated Irregular Networks (TINs). Even though a TIN can admittedly give more accurate results, we opted for the DEM as it significantly reduces the data dimension and processing time. Moreover, in the case of a low point density or a zero-point density patch in the area of interest, the TIN facets become larger in size resulting in unnatural triangular artifacts. DEM uses the interpolation approaches to fill in such void patches. In our method, the TIN input can be easily plugged in if necessary in the place of DEM without requiring any further processing.

For DEM, the grid is overlaid on the terrain surface, so the attribute represents surface elevation obtained by means of interpolation of the corresponding point cloud data. For generating DEM, the spatial interpolation approach generally used are Inverse Distance Weighted (IDW), kriging, Nearest Neighbor etc. We use IDW since it is robust for landscape with high surface variability (Mongus, 2014).

The input to our algorithm was a LAS file, which stores airborne LiDAR data in a binary format. The mean point density of the acquired LiDAR data is $4/m^2$. First, the LAS file for a target area was exported to the ASCII text format and was then read in MATLAB. For rasterization, the small cell size of 2m is chosen to ensure low quantization error. The target area is then overlaid with a 2m x 2m

square grid. The well-known IDW interpolation is then used to predict the voxel value at the center of each cell, using a linearly weighted combination of a set of sample points present in that cell. The weight is a function of the inverse distance from the sample point to the center of the cell. The IDW is again used to interpolate elevation to ‘empty’ cell (cell with no LiDAR point). Here, instead of the neighbor sample point, the neighbor cells value has been used. In Appendix A, the IDW interpolation method is detailed. An example of the resultant square mesh model of a landscape is shown in Figure 31

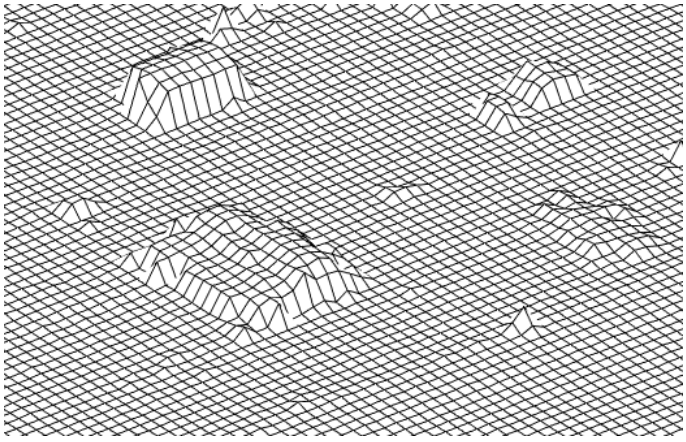


Figure 31 Square mesh model of a landscape

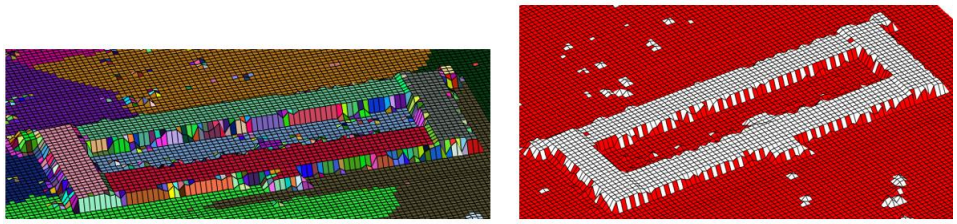
5.2. Pfeifer's Ground Filter

In our work, we selected the well-known surface based filter proposed by Tóvári and Pfeifer (2005) as our pixel-based ground filter. This approach combines planar segmentation and ground filtration using a robust interpolation technique.

In the first step, a plane-fitting algorithm is used to generate planes. The planar segmentation step is based on a region growing algorithm, where the seeds are placed, first randomly, and then in unexplored spaces, to divide the target area into planar segments. Once a seed is selected, the nearest neighbor points are examined, to determine whether they meet certain criteria. Neighbor points that satisfy the criteria are added to the growing region. The details of this algorithm are discussed in Appendix B.

The next step is a robust interpolation. First, planar segments with a size larger than the largest man-made structure possible are extracted and identified as ground. These ground seeds act as ground elevation references to initiate the filtering process. Next, a surface is interpolated initially, using surface moving least squares (MLS), from all points. For each planar segment, a weight is assigned based on the average difference in the distance of the interpolated surface to the constituents observed value. These weights are considered in the next iteration and therefore, segments with a large weight have a larger influence on the run of the surface. This process is iterated until there is no object point left, i.e. their weight becomes zero. The details of this algorithm are discussed in Appendix C.

Figure 32(a) shows the result of the planar segmentation algorithm. Here, each color represents an individual planar segment. Figure 32(b) shows the result after robust interpolation. Here, the red color represents the ground planar segments and the white represents the non-ground planar segments.



(a) Planar segmentation (b) Robust interpolation

Figure 32 Result showing after (a) Planar segmentation (b) Robust Interpolation of a convex object such as complex building

The procedure from (Tóvári and Pfeifer, 2005) was designed to detect planar segments only belonging to conspicuous convex objects such as building, trees etc. As stated in chapter 1, our objective is not only to detect prominent convex objects but also objects that are not so discernable from its background such as mountain and objects that are concave in shape such as a pond, artificial cavities etc. To serve our purpose, we made two modifications in the interpolation method. One modification enables the extraction of planar segments belonging to concave objects. Details of this modification are explained in Appendix C.

The other modification enables the ground filter to detect all possible planar segments belonging to object even those with low heights. Details of the underlying mechanism of this modification are explained below.

As discussed in section 3.3, there are two types of errors that can occur during the filtering process: Type I error and Type II error. The type-I error happens when a ground filter fails to accept a valid bare earth point. Type-II error occurs when it accepts points belonging to an image object as a ground point. In our method, the parameters of the filter are tuned to minimize the type II error i.e. it guarantees the removal of most of the non-ground points, even those that correspond to objects that are small and close to the ground. Therefore, the ground filter is designed to capture every salient point it found on its way during the region growing. Since there is a trade-off involved in making type I error and type II error, type I error increases in the process. However, in the subsequent steps, the extracted object planar segments are scrutinized further and the misclassified ground planar segments are detected and eliminated. Therefore, the combined subsequent stages (Naïve object segmentation and Hierarchical decomposition) act as a Virtual Surveyor based refining module. This reduces the type I error. As a result, both type I and type II error are minimized. Figure 33 shows the flowchart of AVSOE from the perspective where it acts as a ground filter refiner.

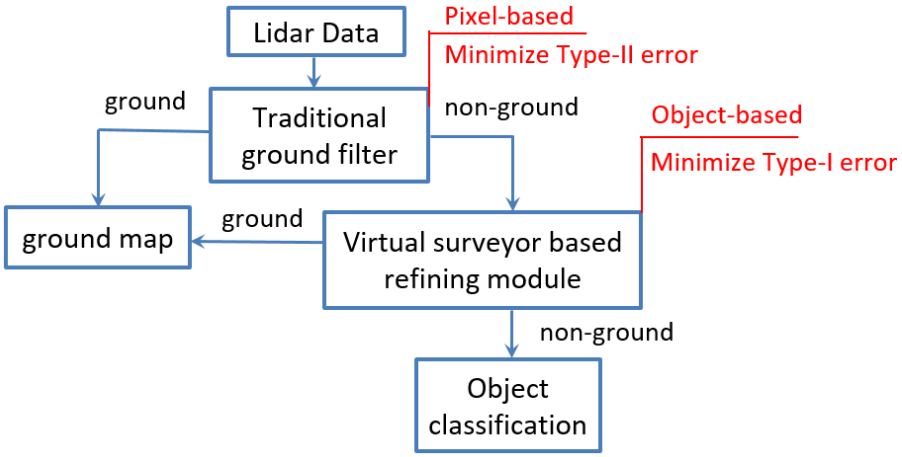


Figure 33 AVSOE as a ground filter refiner

In the next step, the object planar segments extracted using the pixel-based ground filter is grouped using connected component analysis. This is a fast and very simple method to implement clustering. The input planar segment network is first converted to a binary image by assigning those pixels corresponds to object planar segments to 1 and the rest to 0. Here, in this analysis 4-type connectivity is assumed. Figure 34 shows the result of connected component analysis on pixels belonging to a concave object.

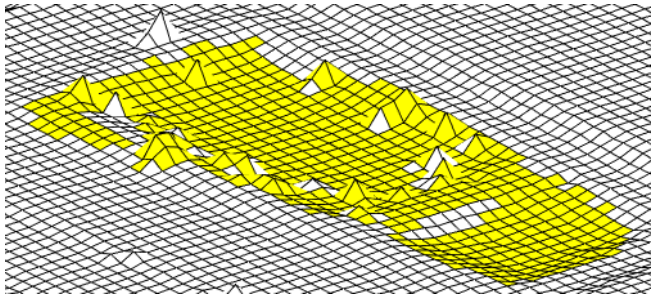
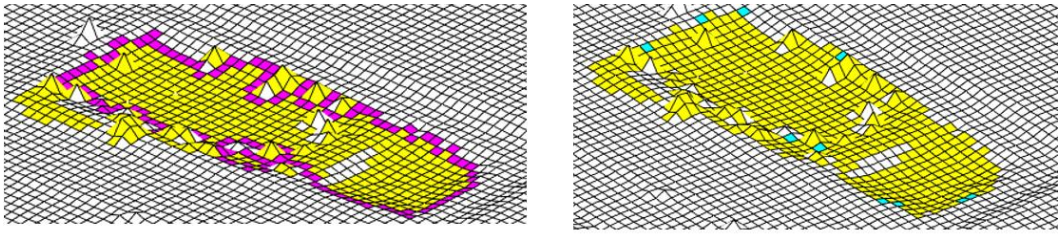


Figure 34 Connected component analysis group pixels (yellow shaded) of a concave-like object

As explained in section 4.3, the naïve object segmentation is a seed growing segmentation method. The seed candidates can be selected from the planar segments of the cluster representing the object. Since it is desired to select the seed candidate from the foothill_slope of the object (according to Section 4.1), the seeds in our method are collected from the border of the cluster. So, we find out the planar segments that lie on the border and designated them as seed candidates. Figure 35(a) shows the estimation of the border of the object shown in Figure 34 and Figure 35(b) shows the seed selected from that border.



(a) (b)

Figure 35 (a) Border selection (magenta) and (b) seed candidate selection (cyan)

Naïve object segmentation sub-algorithm also requires the estimation of the corresponding Background Normal vector of the object. Background Normal vector is the vector perpendicular to the topographic surface onto which the object is placed. To estimate the normal, the planar segments corresponding to the neighboring ground is selected and a plane is fitted to the points using the plane-fitting algorithm described in Appendix B. Figure 36 shows the group of neighboring ground planar segments of the concave object shown in Figure 34.

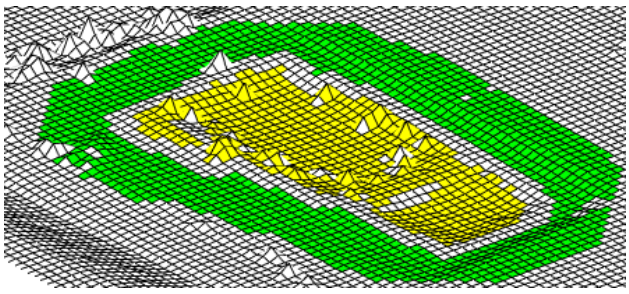


Figure 36 Neighboring ground planar segments (green)

The preprocessing step produces the following: the planar segment network and the object candidates represented by their corresponding seeds and Background Normal vectors (BN vectors). These results are then fed into naïve object segmentation sub-algorithm.

6. Naïve Object Segmentation

Our object segmentation method is inspired by how a human surveyor, placed at somewhere on the foothill_slope of an object would plan her path so that she gradually covers and maps the slope of the entire object without any external help. In essence, this method is presented with the same set of data as the human surveyor and it adaptively analyses the data much like the human surveyor would. Hence, the name Virtual Surveyor. However, the segmentation method only maps the slope surface of the objects leaving behind the top horizontal flat surface of the objects. Objects that lies on the foothill_slope of the parent object are also included in the segmentation as explained in section 4.1. These sub-objects are later detected and segmented out in the hierarchical decomposition stage.

The purpose of the naïve object segmentation algorithm as discussed in section 4.3 is to partition the map of the terrain into regions that contain potential object candidates. This significantly reduces the complexity for the Hierarchical decomposition sub-algorithm to determine the perimeter of the object and its sub-objects. The sub-algorithm considers each of the object candidates extracted from the previous steps to map out a region such that it encompasses the foothill_slope of the object. However, in the process, the segmentation may include nodes (planar segments) that are not part of the foothill_slope of the object. Since the segmentation is far from perfect, the algorithm is termed as ‘Naïve object Segmentation’.

Figure 37 shows the overall flowchart of the AVSOE algorithm with the Naïve object segmentation sub-algorithm in details.

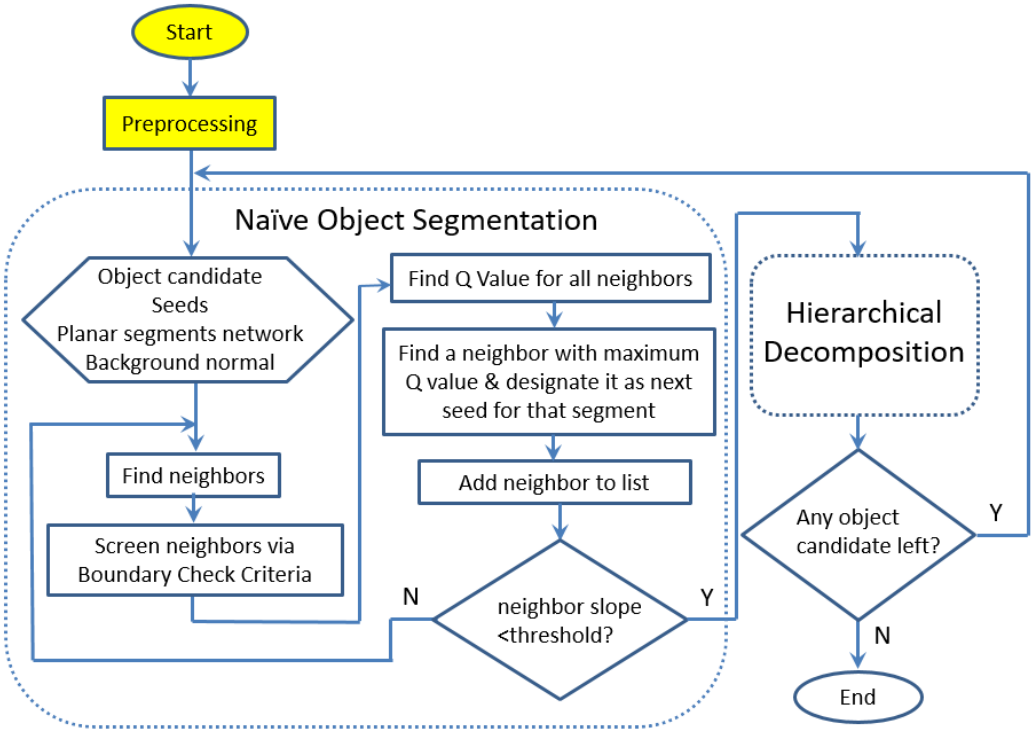


Figure 37 Flowchart of the AVSOE algorithm with the Naïve Object Segmentation stage in details

. The input to the object segmentation module is the planar segments network, object candidates represented by their corresponding seeds location and Background Normal vectors (BN vectors). In the following, the individual parts of the flowchart are explained in detail.

6.1 Segmentation using the on_slope condition

One way to do object segmentation is by using a seed growing segmentation algorithm that starts with a seed at somewhere on the foothill_slope and accepts those neighbors with which the connecting edge obeys the on_slope condition. Theoretically, this would have ensured that the nodes that are being included belong to the foothill_slope of the object. However, in practical, due to the granular representation of the terrain, it is found that segmentation may occasionally cross over the boundary of the object.

To terminate the segmentation, the algorithm computes the magnitude of the Surveyor Guidance vector of the selected neighbor. As can be seen from the Figure 38, the magnitude of the SG vector, which is the cross product of BN vector and SN vector, is proportional to the grade of the slope with respect to the background. So, when the magnitude of the SG vector of the selected neighbor becomes less than a certain low threshold value, this indicates that the segmentation has ventured outside the boundary of the object, so this would prompt the termination of the segmentation process. However, as stated in Section 4.2, the segmentation through on_slope condition, may occasionally cross over the boundary of the object so the termination condition may cause the segmentation to end prematurely, that is, before capturing the entire foothill_slope of the object.

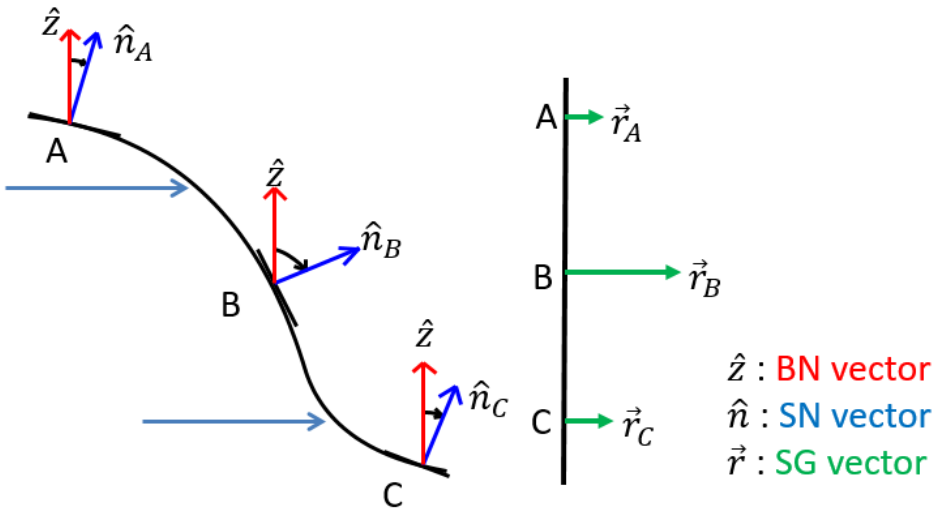


Figure 38 Variation of the magnitude of Surveyor Guidance vector with respect to the profile of the slope

6.2 Segmentation via following the Surveyor Guidance vector

To overcome this problem, we enhance the segmentation strategy. Instead of including all the neighboring nodes that pass the on_slope condition at the same instant, it prioritizes the inclusion of the neighboring node that maximizes the fitness value $Q_{connect}$ which is computed according to this formula (Referring to Figure 25 and Figure 38):

$$Q_{connect} = W_1 \times |first\_dot\_prod| + W_2 \times |second\_dot\_prod| + W_3 \times |\vec{r}_A|$$

eq. 7

Here, the first term measures how close the direction of the path vector is to the Surveyor Guidance vector at the current node, the second term encourages the virtual surveyor to take the path that preserves the SG vector direction and the last term measure the grade of the slope with respect to the background. In this way, the mapping process follows the SG vector and the entire foothill_slope of the object will get mapped before the termination criteria are reached. The neighboring planar segment with the maximum fitness value $Q_{connect}$ is merged into the growing segment region.

However, the fitness equation (eq. 7) faces difficulty in guiding the surveyor around the corner since the *first_dot_prod* measure in that direction is low, as demonstrated with an example in Figure 39. In order to compensate for the low contribution of *first_dot_prod*, we introduce a new measure in the fitness equation called *angle_measure*.

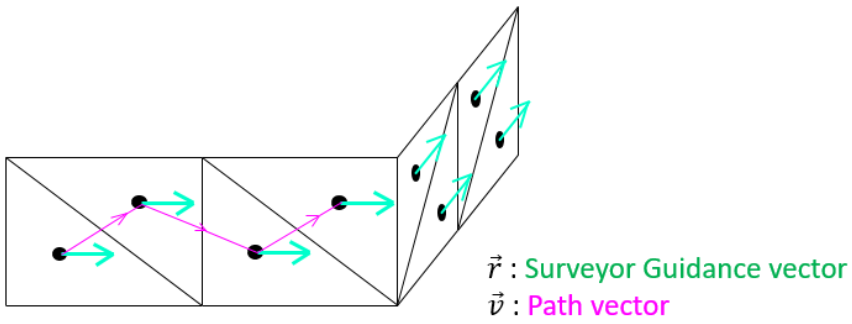


Figure 39 Guiding the surveyor around the corner

Referring to Figure 40 where $\vec{r}_B = \hat{z} \times \hat{n}_B$ and $\vec{r}_A = \hat{z} \times \hat{n}_A$. Here, $\hat{z}$ is the Background Normal vector.

Here, the SG vector corresponds to the current segment (A) and the neighbor segment (B) are first projected onto the horizontal plane

$$proj_{plane}(\vec{r}_A) = \vec{r}_A - (\vec{r}_A . \hat{z}) \times \hat{z} \qquad \textit{eq. 8}$$

$$proj_{plane}(\vec{r}_B) = \vec{r}_B - (\vec{r}_B . \hat{z}) \times \hat{z} \qquad \textit{eq. 9}$$

The *angle_measure* is the angle between the projected SG vectors pair

$$angle\_measure = cos^{-1}\left(\frac{proj_{plane}(\vec{r}_A).proj_{plane}(\vec{r}_B)}{|proj_{plane}(\vec{r}_A)|\times|proj_{plane}(\vec{r}_B)|}\right)$$

eq. 10

It should be noted that angle_measure is impervious to the rotation along the horizontal axis as shown in Figure 40(c). So, the surveyor is encouraged to rotate only along the axis in the direction of BN vector.

With the introduction of the magnitude of angle_measure, the fitness equation as given by:

$$Q_{connect} = W_1 \times |first\_dot\_prod| + W_2 \times |second\_dot\_prod| + W_3 \times |\vec{r}_A| + W_4 \times |angle\_measure| \qquad \textit{eq. 11}$$

Here, W_1, W_2, W_3 and W_4 represent the weights set for each term.

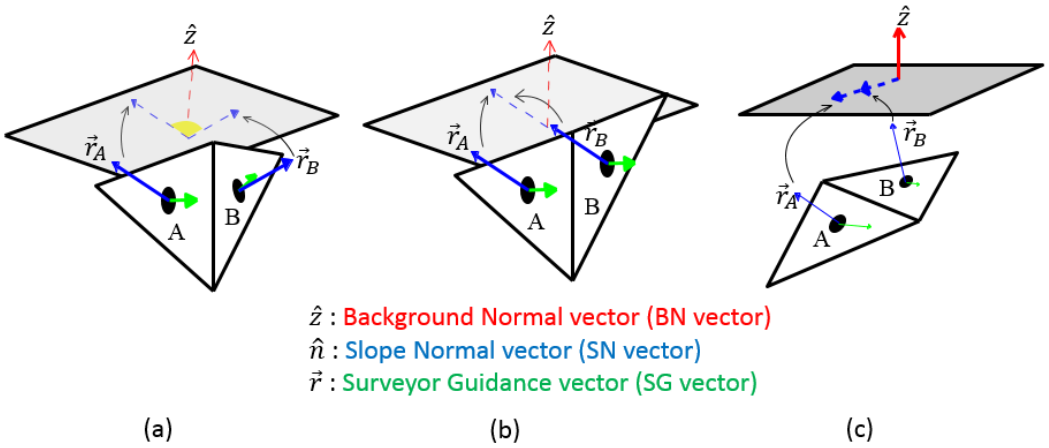


Figure 40 Normal vector projection onto the horizontal plane for adjacent planar segments when they are (a) coplanar when they form a (b) corner about vertical axis and a (c) corner about horizontal axis

Referring to Figure 37, the segmentation algorithm first finds all the nodes that are neighbors to the segment region. Then, it filters out the neighbors that pass the on_slope condition. Next, $Q_{connect}$ is calculated for all remaining neighbors. Next, it finds the neighbor that have maximum $Q_{connect}$ which is then added to the growing region. Table 2 shows the distribution of the weights. The values of the weights were determined empirically.

Table 2 Weight distribution

W_1	W_2	W_3	W_4
35	10	65	20

In order to ensure the naïve segmentation includes the foothill_slope of the object entirely, the threshold value for the termination of the segmentation process is set very low. As stated in section 6.1, the on_slope condition fails in some occasion and the mapping process may cross over the boundary of the object. This cause inclusion of some planar segments that are not part of the object. An example of such false positive planar segments is shown in Figure 41 (b). In the hierarchical decomposition step, the perimeter of the object is correctly detected and the false positive planar segments are consequently removed.

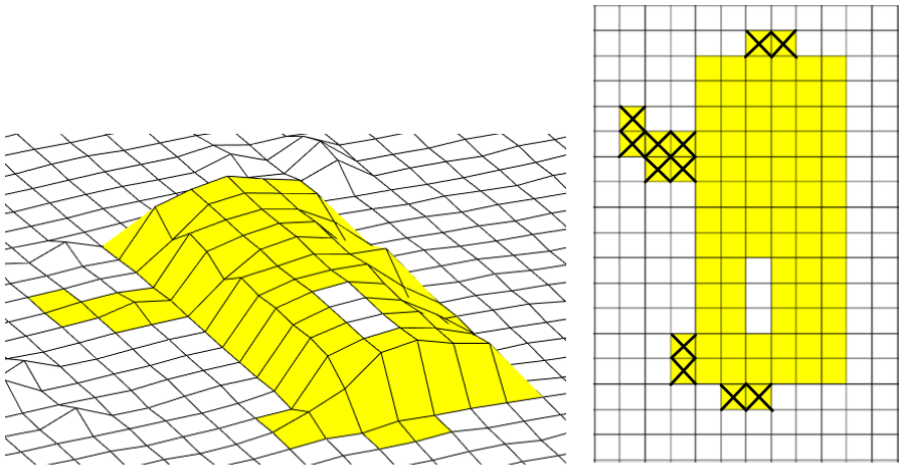


Figure 41 Illustration displaying segmentation result of a house at (a) oblique view, and (b) top view. The false positives planar segments are crossed out

6.3 Segmentation Results

Figure 42 is the segmented result of two adjacent landforms. Here, the segmentation result is expressed in terms of the planar segments. This is a very interesting case, where both the hill and the watershed share the same slope. For this reason, both are segmented from their surrounding using a single seed. In the next step of the hierarchical segmentation algorithm, both will be individually identified.

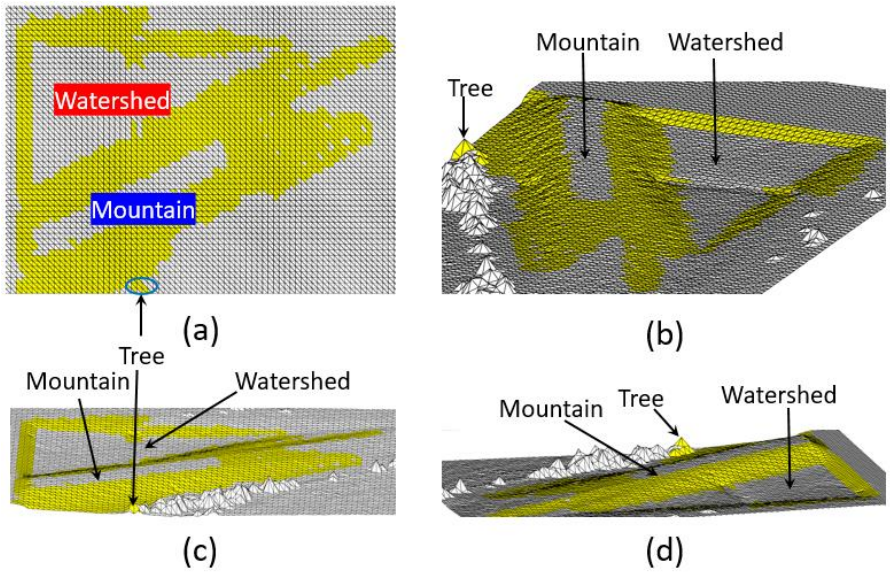


Figure 42 Illustration displaying segmentation result for a hill shaped and watershed shaped landform at (a) top view, and at (b, c, d) different angle views

Figure 43 shows successful segmentation of an artificial pond.

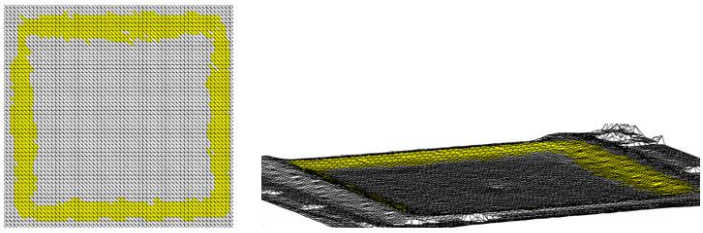


Figure 43 Illustration displaying segmentation result of an artificial pond at (a) top view, and (b) oblique view

Figure 44 shows the segmentation result of a complex building. It is evident from the figure that all the walls of the building have been successfully mapped.

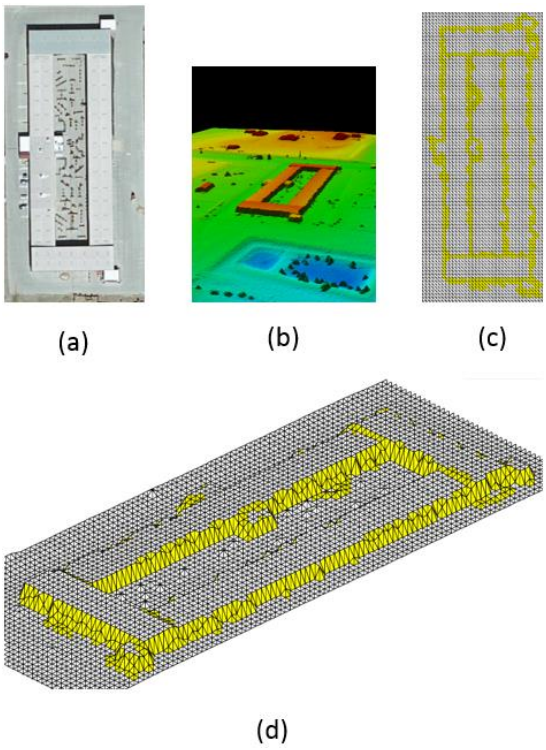


Figure 44 Illustration displaying (a) orthoimage of a complex building, (b) point cloud model of the building, segmentation results at (c) top view, and (d) Oblique view

Figure 45 shows the segmentation result of a pair of adjacent pools. As can be seen from the figure, sub-objects that lie on the foothill_slope of the parent objects are also included in the resultant segment.

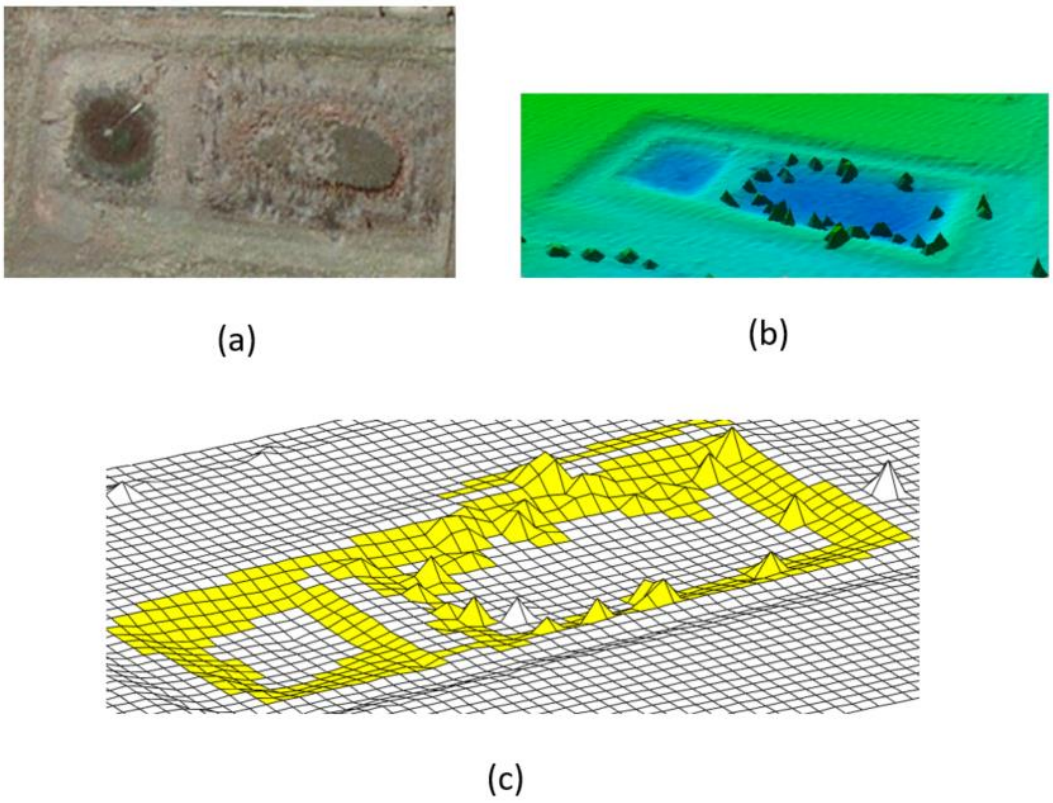


Figure 45 Illustration displaying (a) orthoimage of a pair of adjacent pools, (b) point cloud model of the pools, (c) the segmentation result

7. Hierarchical Decomposition

In the segmentation step, the map is partitioned into regions that each contain an object. The object can be a composite one, containing many sub-objects. The segmentation method ensures the entire foothill_slope of the object and its sub-objects are captured. In the process, false positive planar segments are also included in the naive segment. The next obvious step is to hierarchically decompose the composite object into its constituent objects and draw their boundaries along with the parent object. The extracted sub-object can itself be a distinctive part of the parent object or an independent object. The relationship of the sub-object to the parent object will be clarified in the Hierarchical decomposition step. Figure 46 shows the overall flowchart of the AVSOE algorithm with the Hierarchical Decomposition stage in details. In the following, individual parts of the flowchart of the Hierarchical Decomposition stage are explained in detail.

The Hierarchical decomposition sub-algorithm work on the segmentation result delivered by the Virtual Surveyor based Naïve Segmentation algorithm. The purpose of the VS Hierarchical Decomposition is twofold:

1. Identify the constituent sub-objects of the composite object
2. Find the boundaries associated with the parent object and each of its sub-objects.

In other words, the hierarchical decomposition process segments out the object and its sub-objects. As shown in Figure 46, this segmentation of object and

its sub-objects is carried out in a recursive pattern. As stated in section 4.3, the sub-algorithm first constructs a directed graph inside the naïve segment where the nodes are the planar segments and the edges obey the path_loop condition. This in-segment graph is termed as ‘complete_flow_graph’.

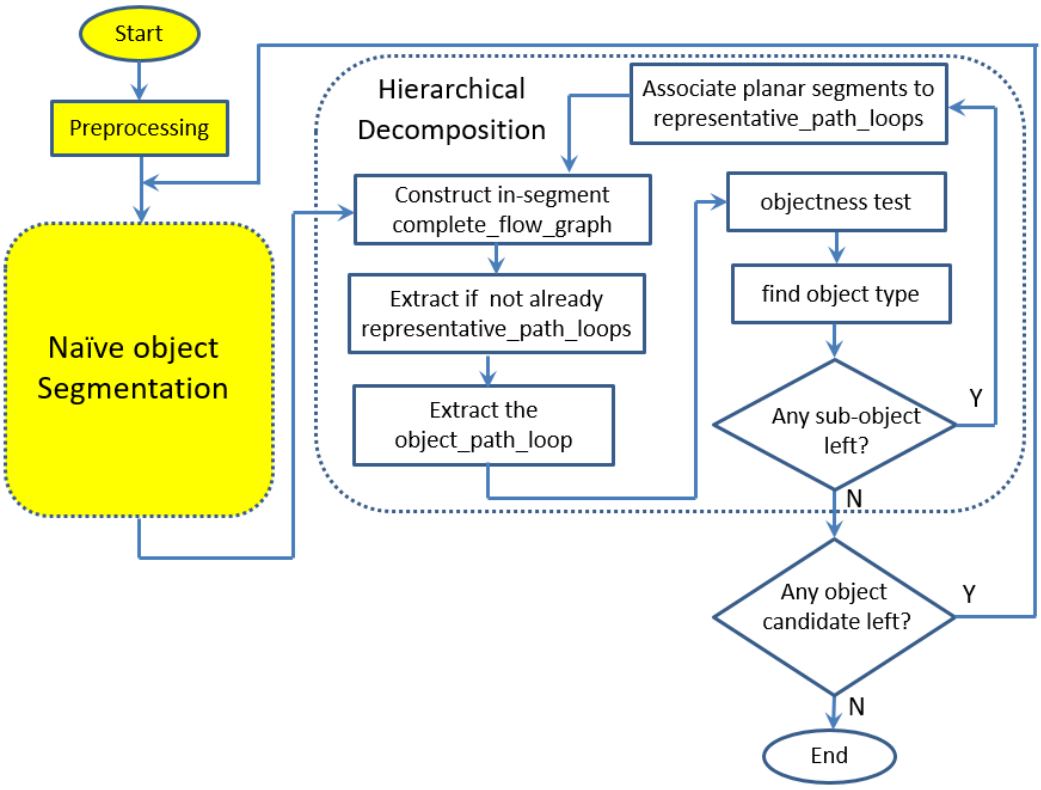


Figure 46 Flowchart of the AVSOE with the Hierarchical Decomposition stage in details

7.1 Generating the complete_flow_graph

As stated in Section 4.3, in any convex or concave-like object, there exists a set of circular path loops, associated to its foothill_slope, which encircles the entire structure. For each point on the slope, there exists at least one member of that set of path loops, which passes through the point. The normal at each point of the circular path loop point inward or outward.

In section 4.2, the following conclusions were drawn:

1. Each object in a topographic map can be represented by a set of circular path_loops where the dot product between their Path vector and the Surveyor Guidance vector (SG vector) at each point has the same sign i.e., the loop follows the path_loop condition.
2. Every point on the object can be connected to any path_loop via a chain of connecting edges that satisfies the on_slope condition.

From these conclusions, we derived the idea of constructing a graph, where each planar segment inside the object segment is connected to its neighboring planar segments if it follows the path_loop condition. The graph would capture not only all the possible path_loops but also the chain of connecting edges that connect each planar segment constituting the naïve segment to any of that loop. We call this graph '*complete_flow_graph'*. A real-world example of such a graph is shown in Figure 47 and Figure 48.

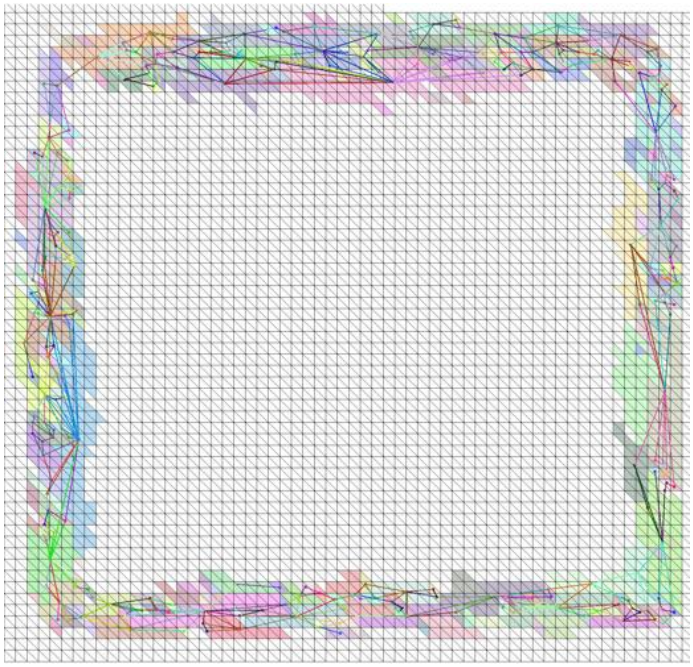


Figure 47 Illustration of the complete_flow_graph of the artificial pond (*Figure 43*)

During the formation of the complete_flow_graph, the procedure also computes the $Q_{connect}$ value for each connection. The mathematical expression for evaluating $Q_{connect}$ value is shown in eq. 2. This $Q_{connect}$ value of the edge quantity the likelihood of the neighboring nodes belonging to the same foothill_slope. These $Q_{connect}$ values for each connection are recorded in a map referred to as $Q_{connect}$_map which is later used in subsequent steps.

The $Q_{connect}$ formulae and the weights assigned is restated below:

$$Q_{connect} = W_1 \times \left|first_{dot_{prod}}\right| + W_2 \times \left|second_{dot_{prod}}\right| + W_3 \times |angle_{measure}| + W_4 \times |\vec{r}_A|$$

where, $W_1 = 45$, $W_2 = 10$, $W_3 = 40$ and $W_4 = 5$

Here, the weight is assigned such that high preference is given to neighbors which lie in the direction close to the SG Vector. This encourages a chain of connections to form around objects.

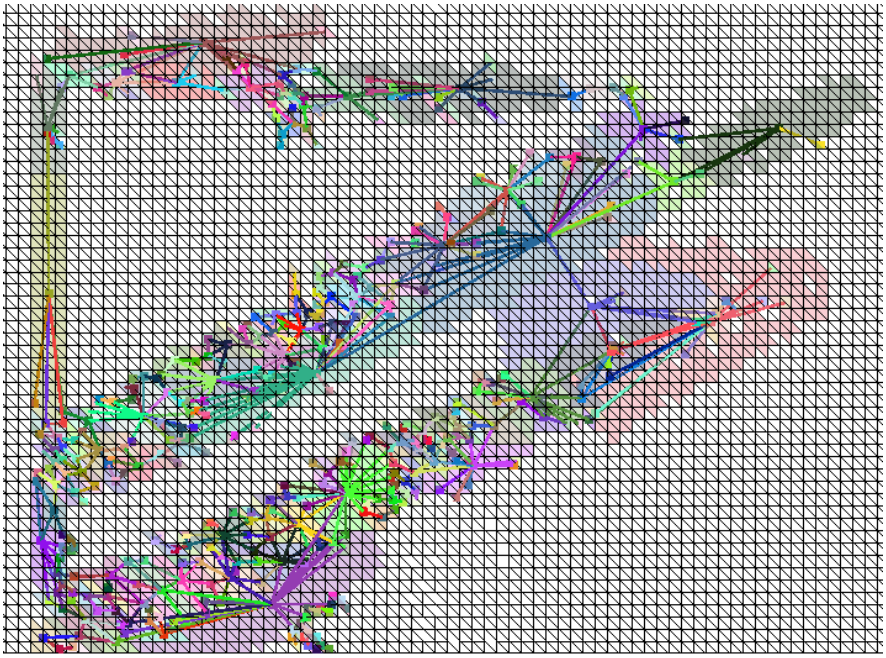


Figure 48 Illustration of the complete_flow_graph of a composite object (hill shaped, watershed shaped landform and a tree) (Figure 42)

As is evident in the figure above, multiple numbers of path_loop encircle each object and sub-object present in the scene. As stated in section 4.3, detection of these path_loops will reveal simultaneously, the presence and position of all objects and sub-objects in the scene. However, ambiguities may arise due to the multiplicity of path_loops associated with each object. A sub-algorithm is designed to tackle this problem. The algorithm deals with the multiplicity problem by merging

parallel loops to a single loop that represent the group. Hence, the loop is called '*representative_path_loop'*.

7.2 Extracting representative_path_loops

Referring back to Figure 46, the flowchart for the algorithm that extracts path_loops is shown in Figure 49.

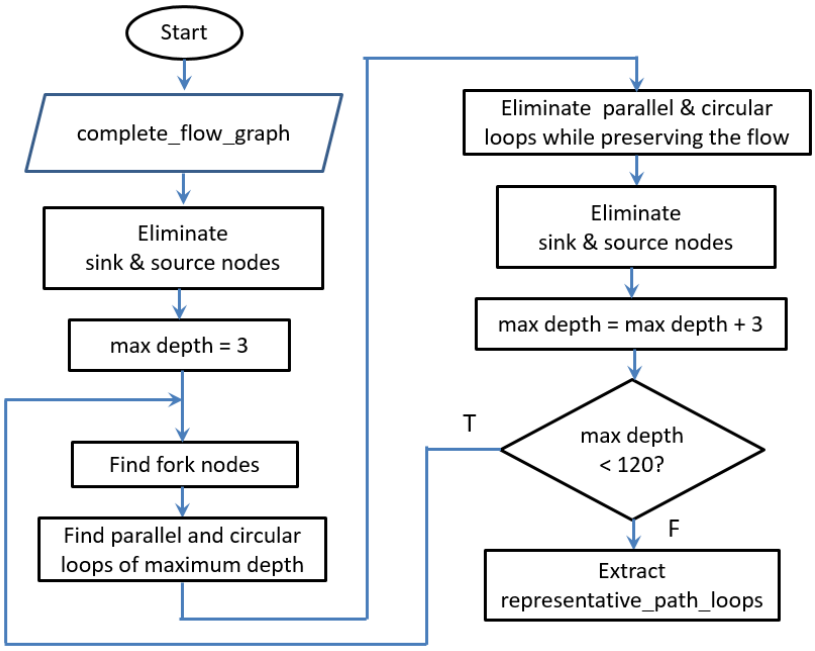


Figure 49 Flowchart of the sub-algorithm that extracts path_loops

Our objective for this algorithm is to extract from each naïve segment a single path_loop that represents the set of circular path_loops encircling each object. We refer to this special path_loop as '*representative_path_loop'*, which

enables detection of all the convex or concave-like sub-objects present in the naïve segment. Moreover, they provide information about the hierarchical and neighborhood relationship among the objects and sub-objects. In the following, individual parts of the flowchart are explained in detail.

7.2.1 Eliminating the sink and source nodes

In the subsequent chapter, the term ‘flow’ has been used frequently. Flow is simply the traversal of the directed graph. In the complete_flow_graph, there are some planar segments (nodes of the graph), where the flow stops and others where the flow starts. Neither set is part of the closed loops that we desire to extract. The nodes that do not connect to any other nodes are sink nodes or sink planar segments, whereas the nodes that are not connected by any other nodes are source nodes or source planar segments. Figure 50 shows a map of the sink and source planar segments of a given naïve segment. The objective of this sub-algorithm is, therefore, to remove all connecting edges that lead to the sink nodes and all connecting edges that connect source nodes to the complete_flow_graph.

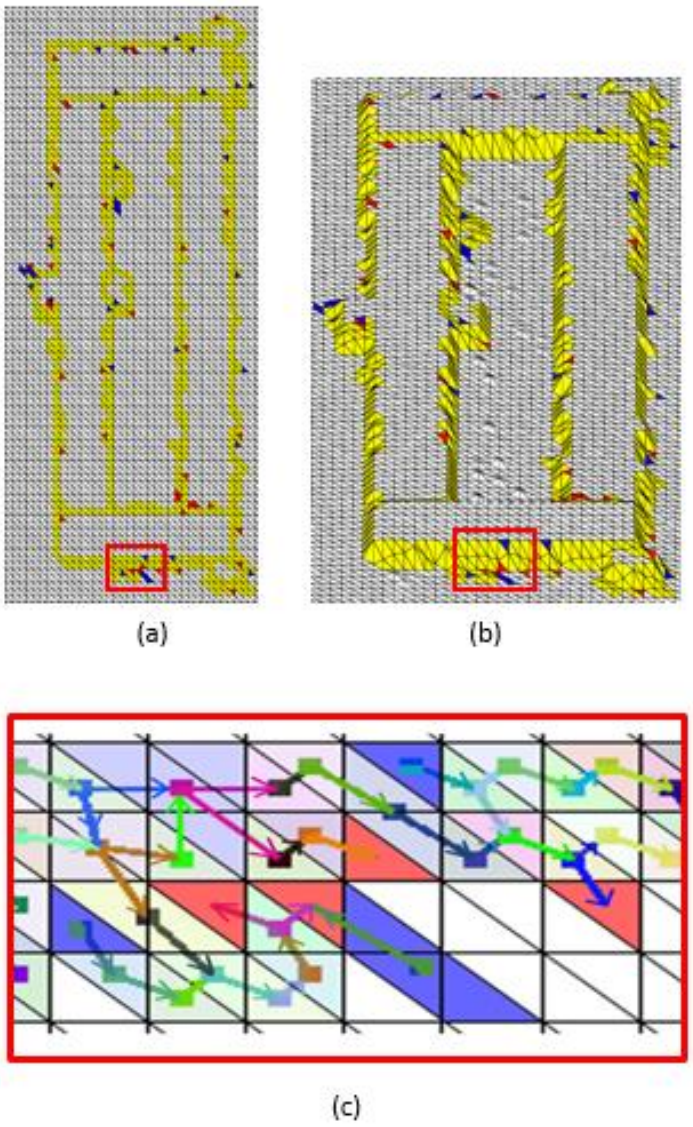


Figure 50 Sink (red) and source (blue) nodes found in the object segment (Figure 44) depicted in a (a) top view, a (b) oblique view and a (c) zoomed-in view of the enclosed portion by the rectangular red box shown in (a) with connecting edges

The flowchart of the sub-algorithm that eliminates source and sink nodes are shown in Figure 51

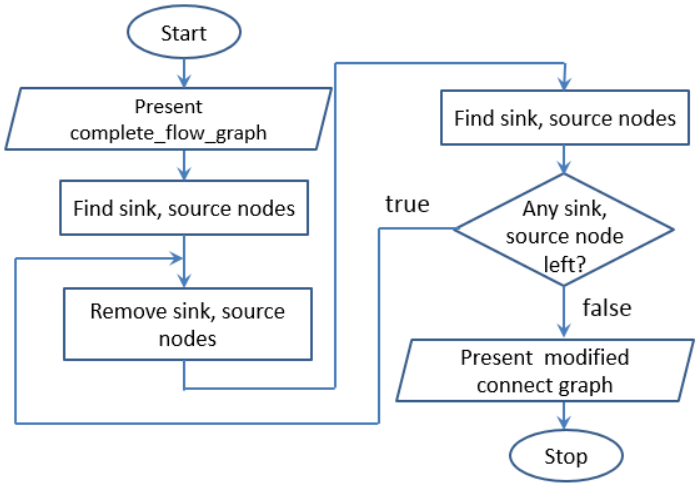


Figure 51 Flow chart of the sub-algorithm that eliminate the source and the sink nodes

The input to this algorithm is the complete_flow_graph generated from the previous step. The operation begins with detection of the sink and the source nodes in the graph and then they are removed along with their connections. Consequently, new sink and source nodes are generated. This pruning process continues in a loop until no sink and source nodes are left in the graph.

7.2.2 Finding the fork nodes

A complex object can consist of many sub-objects and their corresponding path_loops may share a section of the loop. The segments representing the start point and the end point of the common section between a pair of path_loops are referred to as fork segments. We define two types of fork segments: the in-fork segment and the out-fork segment.

Here, the in-fork segment is one, where there are at least two incoming paths and one outgoing path (a simulated example is shown in Figure 52(a)) whereas, an out-fork segment is one, where there are at least two outgoing paths and one incoming path (Figure 52 (b)).

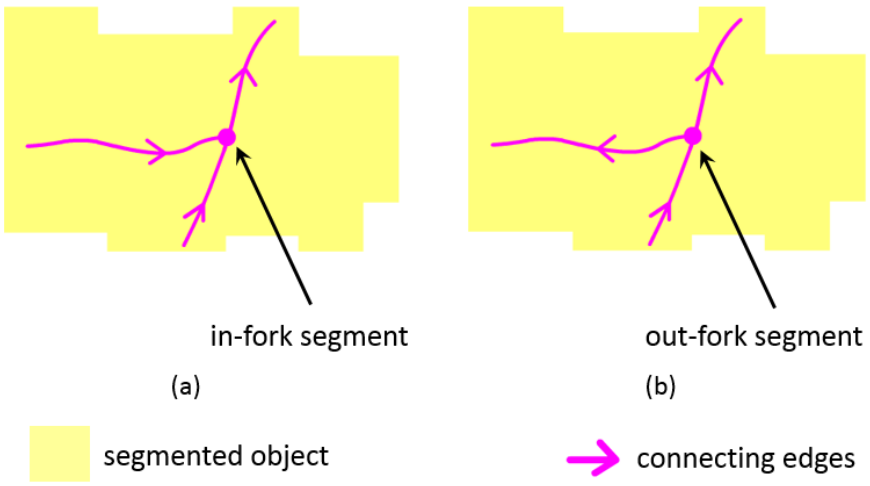


Figure 52 Stylistically simulated example of (a) out-fork segment (b) in-fork segment

In a complex object structure, both types of fork segments can be found. Figure 53 shows the major path loops that can be found in the complex object structure shown in Figure 44.

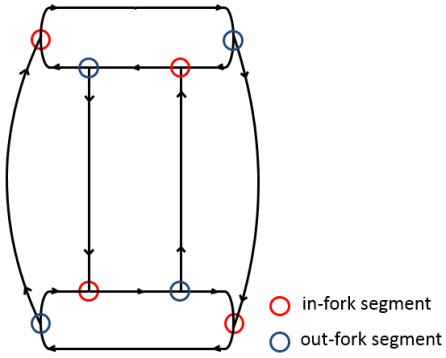


Figure 53 Location of the in-fork and out-fork segments in the complex conjoined path loops formed around object shown in Figure 44

7.2.3 Finding the parallel and the circular loops of maximum depth

We defined two types of loops: circular loop (path_loop) and parallel loop. For a circular loop, a surveyor following the directed graph would return to her initial position. A stylistically simulated example of a circular path loop is shown in Figure 54.

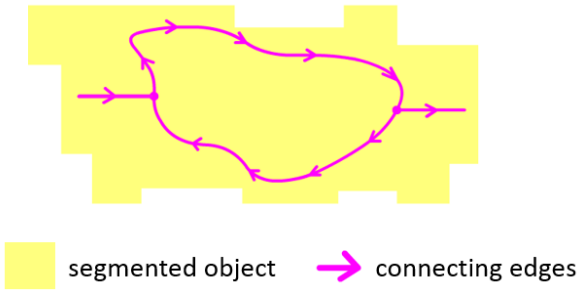


Figure 54 Stylistically simulated example of a circular loop

For a parallel loop, there is more than one path a surveyor can take, which will eventually lead to the same destination. We call each of these paths 'loop_arm'. An arm that branches from an out-fork node in the right-hand direction is termed '*right arm'*, and the one that branches in the left-hand direction, the '*left arm'*. A simulated example of a parallel loop is shown in Figure 55.

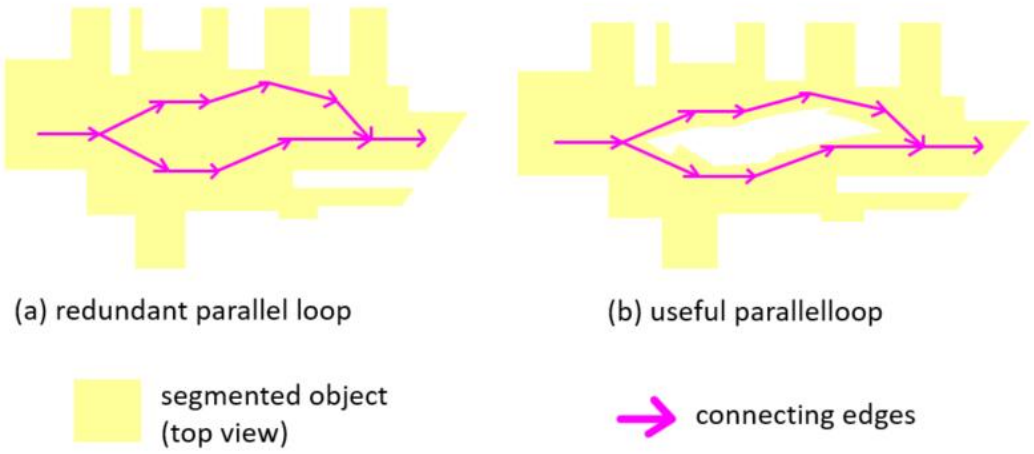


Figure 55 Simulated example of (a) a redundant parallel loop and (b) a useful parallel loop

Most of the parallel loops formed in the complete_flow_graph are redundant in the context of flow around the object. We refer to them as redundant parallel loops. But in certain circumstances, such as when the objects (of different height) are adjacent to each other and there are no planar segments separating them, the parallel loop would be formed and should be considered to detect sub-objects. It is then called a ‘useful parallel loop’. One such example is shown in Figure 56. The redundant parallel path loops are defined as the ones, which do not encompass the object and therefore there exist no unmapped region within the path loop as demonstrated in the simulated example shown in Figure 55. This distinguishing feature is used by a procedure to seek out useful and redundant parallel loops. The extracted useful parallel loop can be converted to the desired circular loop as demonstrated in Figure 57.

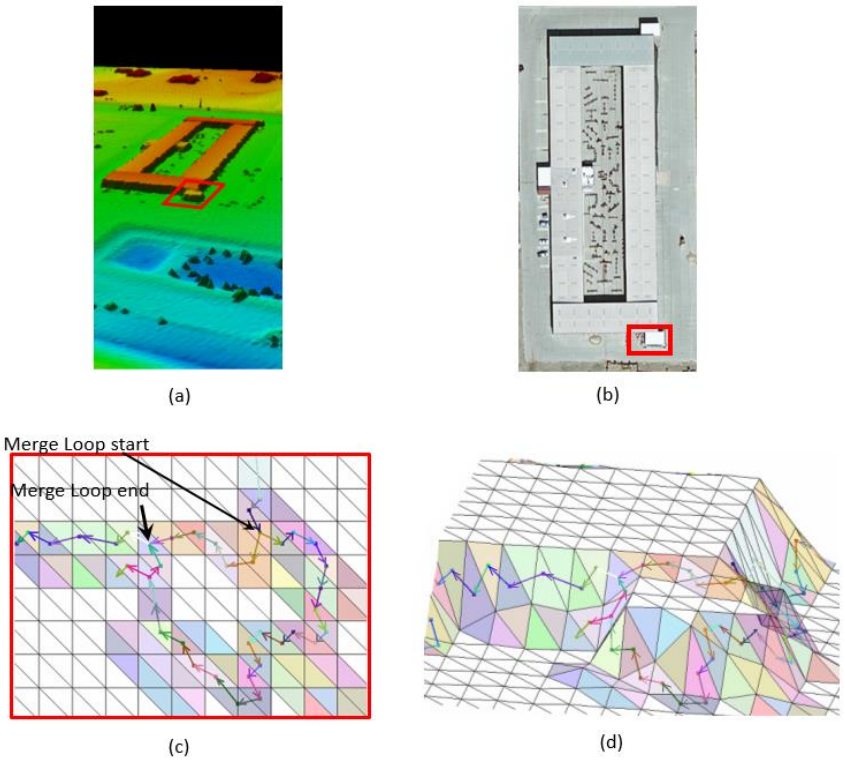


Figure 56 Illustration displaying a (a) 3D point cloud model of the target area which contains a complex object, a (b) orthophoto of the building, (c) top view of the parallel loop formation focusing the area in the red bounding box drawn in (b), and (d) oblique view of the area in the red bounding box.

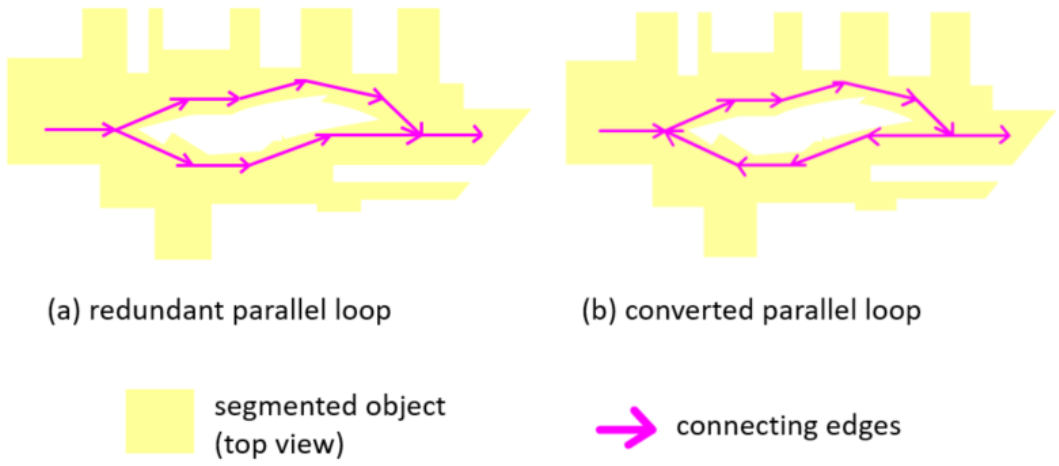


Figure 57 Simulated example of a (a) redundant parallel loop, and converted (b) circular loop

Figure 58 shows the flowchart of the sub-algorithm employed to extract all the parallel loops and the path_loops. This algorithm is analogous to the Depth First Search (DFS) algorithm used in finding cycle graph from a directed graph.

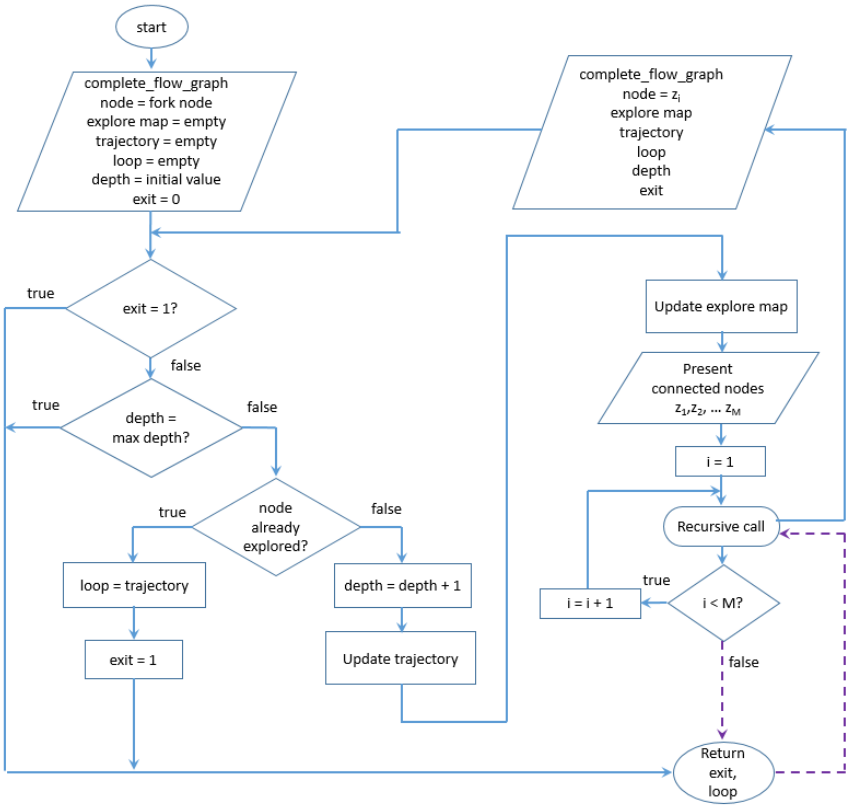


Figure 58 Flowchart of a sub-algorithm which extracts all merge loops and circular loops of a given max depth

The upper limit for the size of loops to be extracted is given as input to the sub-algorithm. This input variable is referred to as ‘max depth’. The other inputs to the functions are the complete_flow_graph and a fork node. In a dance graph, almost all the path_loops contain one or more fork nodes. All the fork nodes present in the object segments are extracted in the previous step of the flowchart as shown in Figure 49. The loop extractor procedure considers each of these fork nodes and explores the flow graph starting from the considered fork node in a

recursive fashion. The recursion ceases to “wind up” and starts to “unwind” once the depth of the tree explored reaches the max depth. Throughout the process, the procedure maintains a map that keeps track of the nodes that have already been explored. This map is referred to as the ‘*explored map*’. In the case of the parallel loops as well as the path_loops, a node would eventually get revisited during the traversal. The procedure detects the event with the aid of the explored map and initiates the exit of the process flow from the recursion cycle. The chain of connecting edges which leads to such situation is recorded and returned to the function caller.

7.2.4 Eliminating the parallel and the circular loops

In this sub-algorithm, the primary objective is to extract all the circular path_loops and useful parallel loops from the complete_flow_graph. Depending on the size of the convex/concave-like sub-objects present in the naïve segment, the loops can exist in various lengths. For a composite object, the largest circular loop can be found around the main parent object. In the high connectivity graph, such as complete_flow_map, the extraction of large loops using the method described in the previous section would take increasingly high computational time. It is, therefore, imperative to simplify the graph before running the path_loops extraction algorithm with a large max depth input. In the flowchart shown in Figure 49, there is a program control flow loop, where during each iteration the max depth is increased from a small value and, the parallel and circular loops of length less

than or equal to that of max depth are extracted and then they are gradually removed from the graph, while making sure that the overall flow remains undisrupted. The simplification of the graph enables the loop extraction of the successive larger length in a very short period of time.

A simple simulated example of the elimination process is demonstrated in Figure 59 for the circular loop and in Figure 60 for the parallel loop. In both cases, a connecting edge constituting the loop is strategically detached. This neutralizes the loop while the flow (from A to B) remain undisrupted. However, this leads to the formation of a redundant arm with a loose end (sink or source node) as shown in the figure. Such loose end arms produced are later cleaned up by the sink and source nodes eliminator stage (See the flowchart in *Figure 49*)

The simulated example is shown in Figure 59 and Figure 60 is a simple example but the graph can get more complicated than this. Figure 61 shows an example of a parallel loop with various complex configurations.

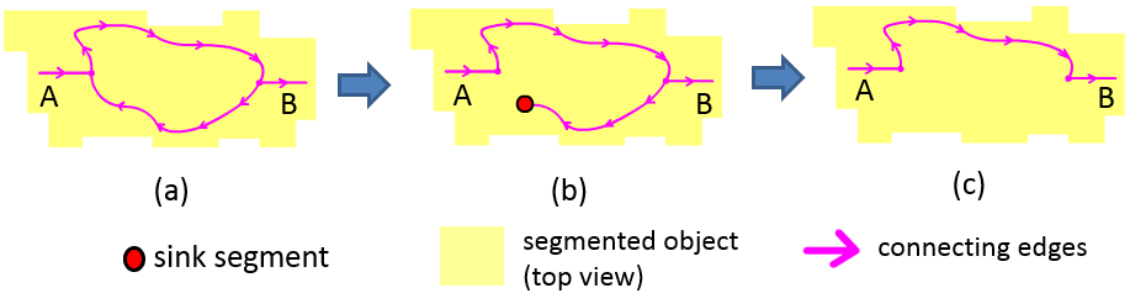


Figure 59 An example of a (a) circular loop, the (b) circular loop after detachment and then the (c) circular loop after cleanup

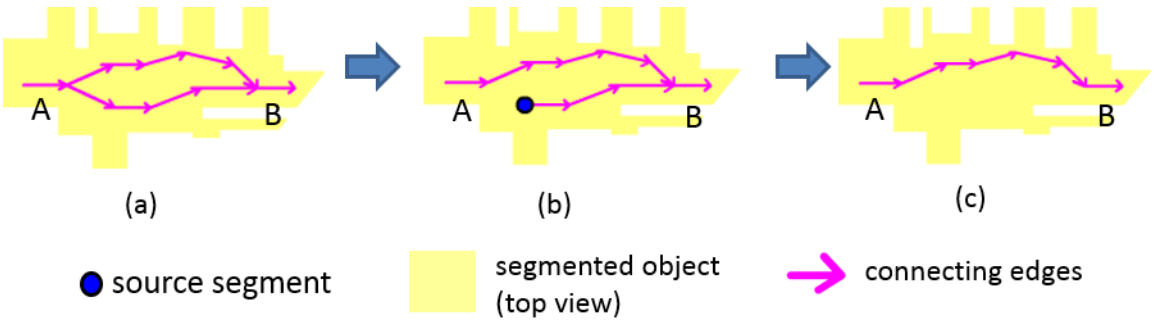


Figure 60 An example of a (a) parallel loop, the (b) parallel loop after detachment and then the (c) parallel loop after cleanup

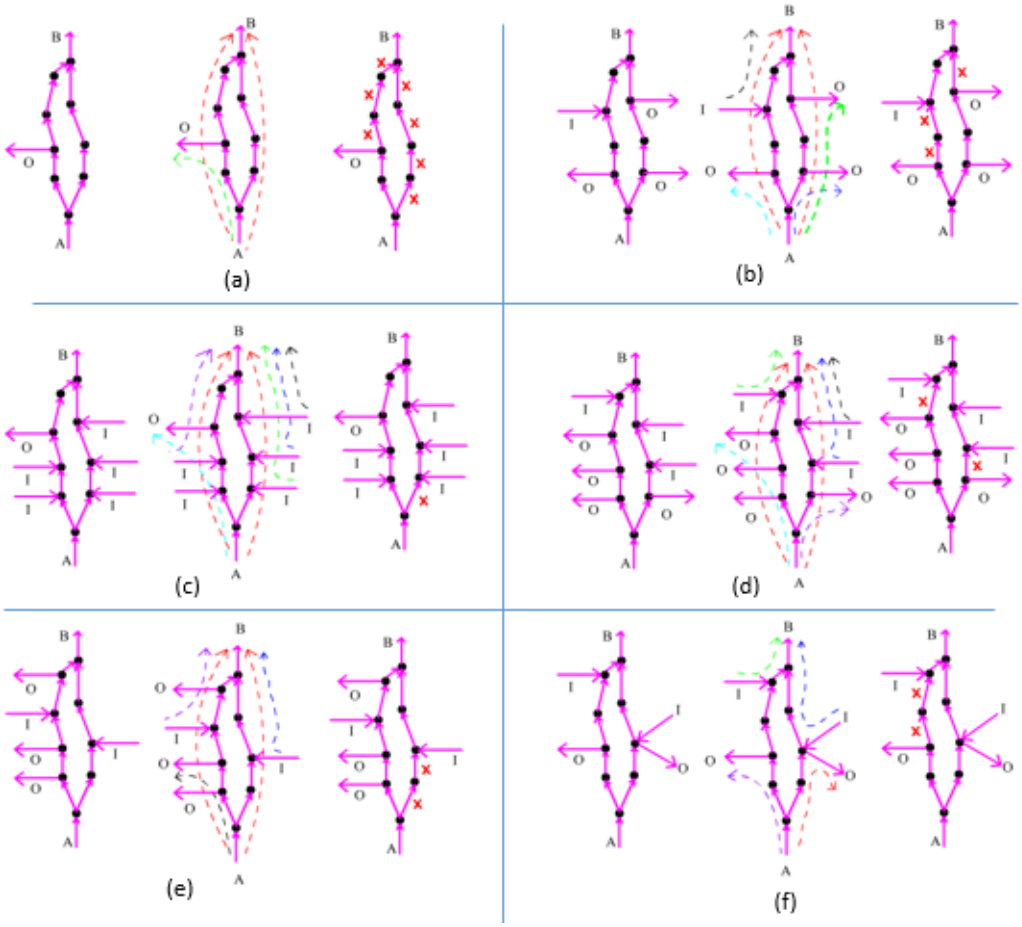


Figure 61 Six different configurations of parallel loops, each showing the path flow and the connecting edge candidates to be clipped

Here in the figure, *I* stand for the in-fork nodes and *O* stands for the out-fork nodes. The connecting edge candidates to be clipped are marked by red crosses. For each configuration, there can be several such candidates. Only one needs to be selected to render the loop inoperable. Whatever the choice is, none of the flow in the configuration will be compromised.

By examining various possible configurations possible for a parallel loop, a set of heuristic rules is derived for the parallel loops to determine the connecting edge candidates that are safe to be clipped. The rules are given below with reference to Figure 61.

1. If there exists a node in the path loop_arm which is both an in-fork and an out-fork, then ignore. (See Figure 61(f) right arm)
2. If the arm contains no fork node, then every connecting edges of that arm is a candidate. (See Figure 61(a) right arm)
3. From A to B, if the configuration is such that the fork nodes in the arm are sequenced *OOOOO... O* then the connecting edge candidates for clipping are present in the configuration as *OOOOO... OXX…X, where X represents the candidates i.e. all connecting edges in the arm after the final out-fork node in the sequence,* are eligible candidates. (See Figure 61(a) right arm, (b) left arm)
4. From A to B, if the configuration is such that the fork nodes in the arm are sequenced *IIIIIIIII…. I* then the connecting edge candidates for clipping are present in the configuration as *XX...X IIIIIIIII…. I*. (See Figure 61(a) right arm, (b) right arm)
5. From A to B, if the configuration is such that the fork nodes in the arm are sequenced *OOO... O IIIIII…. I* then the connecting edge candidates for clipping are present in configuration as *OOO... OXX...X IIIIII…. I*. (See Figure 61(b) left arm, (d) right arm, left arm, (f)left arm)
6. The arm configuration that doesn't satisfy the above stated (2, 3, 4, 5) condition, are ignored. (See Figure 61(c) left arm, (e) left arm)

If connecting edge candidates for clipping are found in both arms of the parallel loop, then the sum of $Q_{connect}$ values for both arm is calculated and then their Q average value ($\bar{Q}$) is evaluated by dividing the sum with the number of connecting edges in the arm. The arm with the lower Q average value is chosen for clipping. This ensure that the chain of connecting edges that remain intact are the best among its peer.

Figure 62 shows an example of circular loops with various complex configurations.

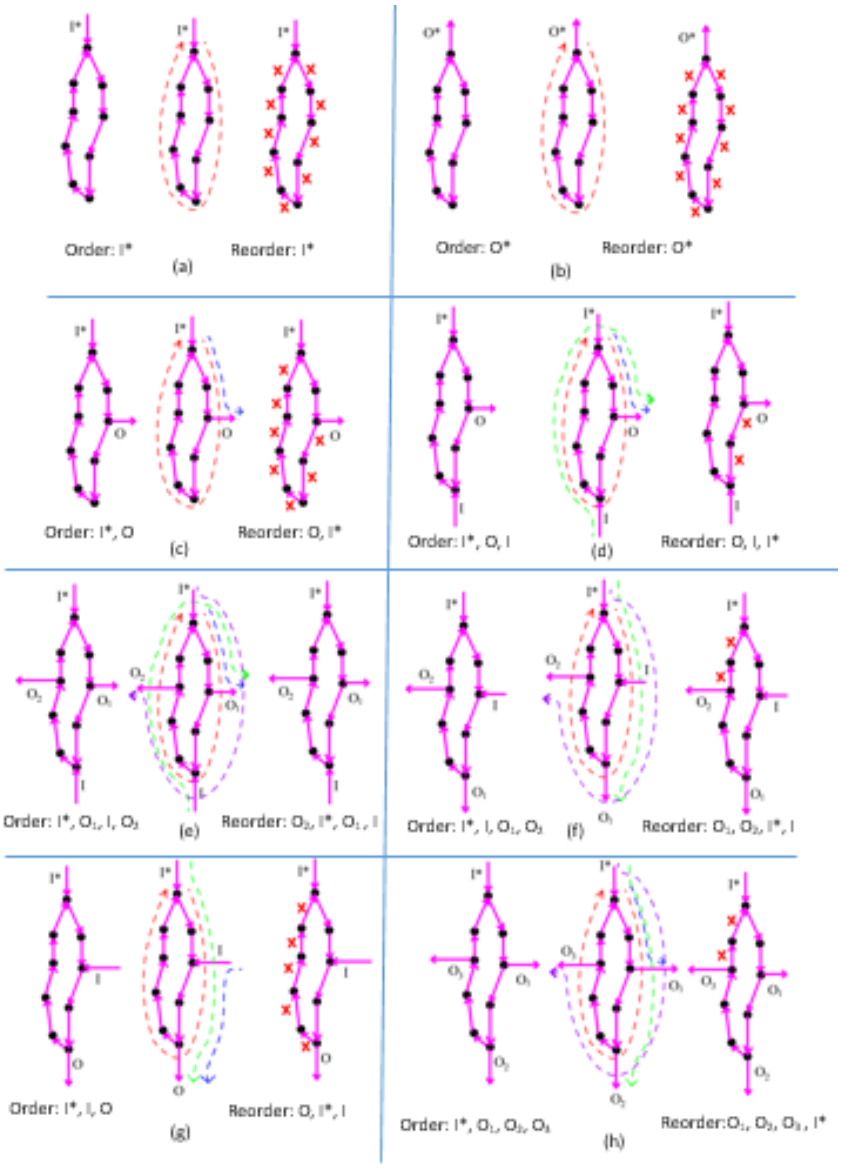


Figure 62 Six different configurations of circular loops, each showing the flow and the connecting edge candidates to be clipped

Here also, for each configuration, there are several options but just one needs to be taken to achieve the purpose. Whatever the choice is, none of the path flow in the configuration will be compromised. Here, the sequence of the fork node in the path_loop is ordered in the direction of path flow starting with in-fork node I*. This sequence is recorded in an array, which is later used to find the connecting edge candidates that can be clipped.

By examining various possible configurations for the circular loop, a set of heuristic rules is derived for the circular loops to determine those connecting edge candidates. The rules are given below with referring to Figure 62.

1. If the configuration in the path loop is such that there is no fork node present, then all connecting edges forming the loop are eligible candidates. (See Figure 62 (a, b))
2. If the configuration in the path loop is such that the fork nodes are all the same type i.e. all are either in-fork nodes or out-fork nodes, then all connecting edges forming the loop are eligible candidates.
3. If the configuration in the path loop is such that there exist a mixture of in-fork and out-fork nodes, then a procedure is assigned to circularly shift the elements of the sequence array mentioned above. The shifting continues until the first element becomes an out-fork node and the last element is an in-fork node. The reordered sequence becomes interesting when the in-fork and out-fork nodes can be divided into two clearly separate groups i.e. if the sequence is like *OOOO…OIIII…I*. Then the connecting edge candidates for clipping are present in configuration as *OOO... OXX...X IIIIII…. I*. i.e. all connections between the

last out-fork node and the first in-fork node in the sequence. (See Figure 62 (c, d, f, g, h))

4. If the loop configuration doesn't satisfy the above stated (1, 2, 3) conditions, then no further action is required. (See Figure 62 (e))

7.2.5 Presenting the representative_path_loops

The program control flow loop (Figure 49) begins with a low-value max depth. Inside the program control flow loop, the path loops of size less than or equal to the max depth are extracted and removed. The extracted circular path_loops and the useful parallel loops are recorded. Removing the path_loops and parallel loops simplifies the graph so that after each successive iteration with a little higher max depth, the graph can be explored in a very short period to extract the loops. Once the value of max depth reaches a preset higher value, it is safe to say that all the path loops present in the object image segment are extracted. The gradual removal of parallel loops ensures that one circular path_loop among the set of circular path_loops survives per object i.e. out of many path loops, the one with the highest Q average value ($\bar{Q}$) remains intact. Therefore, each of these extracted path loops acts as a representative_path_loop for its object. Each representative_path_loop reveals the presence and the position of all the sub-objects. A simulated example of a complete_flow_graph in an object segment is shown in Figure 63(a). The final simulated result of the path loops grouping algorithm is shown in Figure 63(b). The representative loops of the sub-objects are connected to the representative loop of the main parent object. This indicates the hierarchical relationship between objects and their sub-objects.

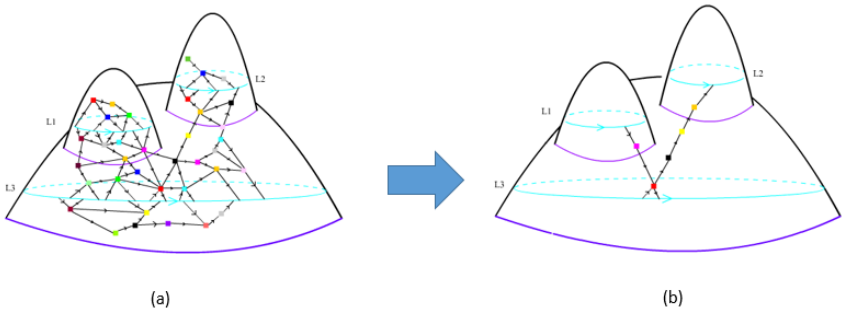


Figure 63 Illustration displaying a (a) complete_flow_graph, and the (b) extracted representative_path_loops

Figure 64 shows the representative_path_loop, extracted from the complete_flow_graph of an artificial pond, shown in Figure 47. Figure 65 display the representative_path_loop, extracted from the complete_flow_graph of a composite object (hill shaped, watershed shaped landform and a tree), shown in Figure 48. Here, the hierarchical relations among sub-objects can be observed.

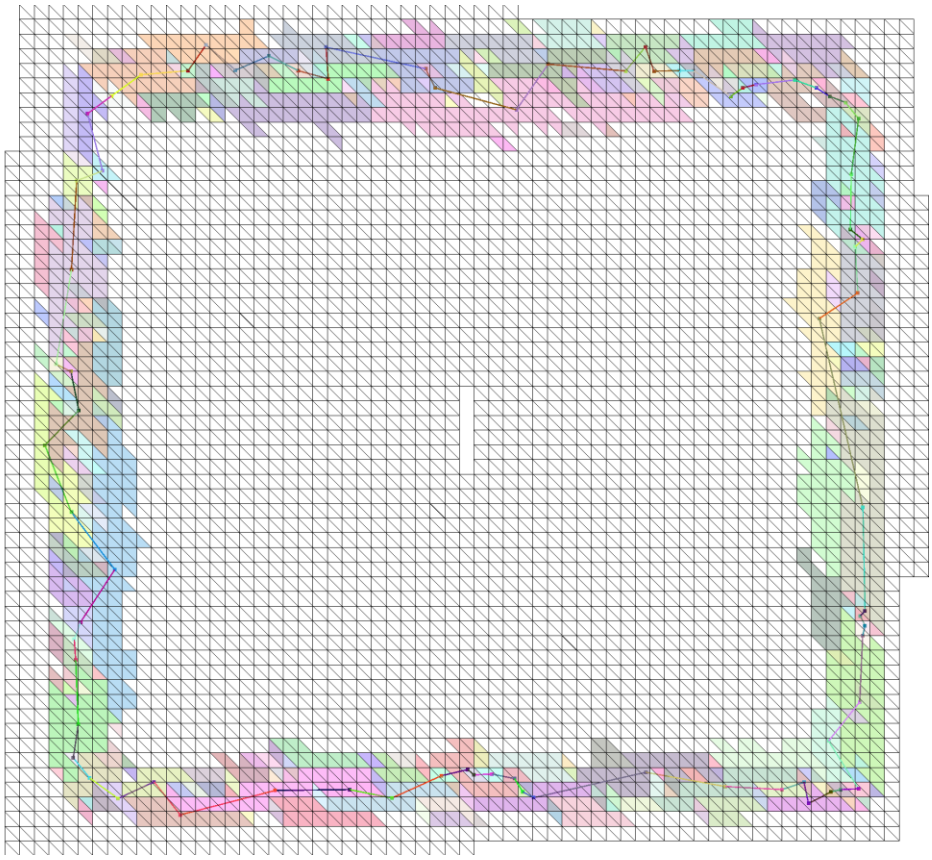


Figure 64 Extracted representative_path_loop of the artificial pond

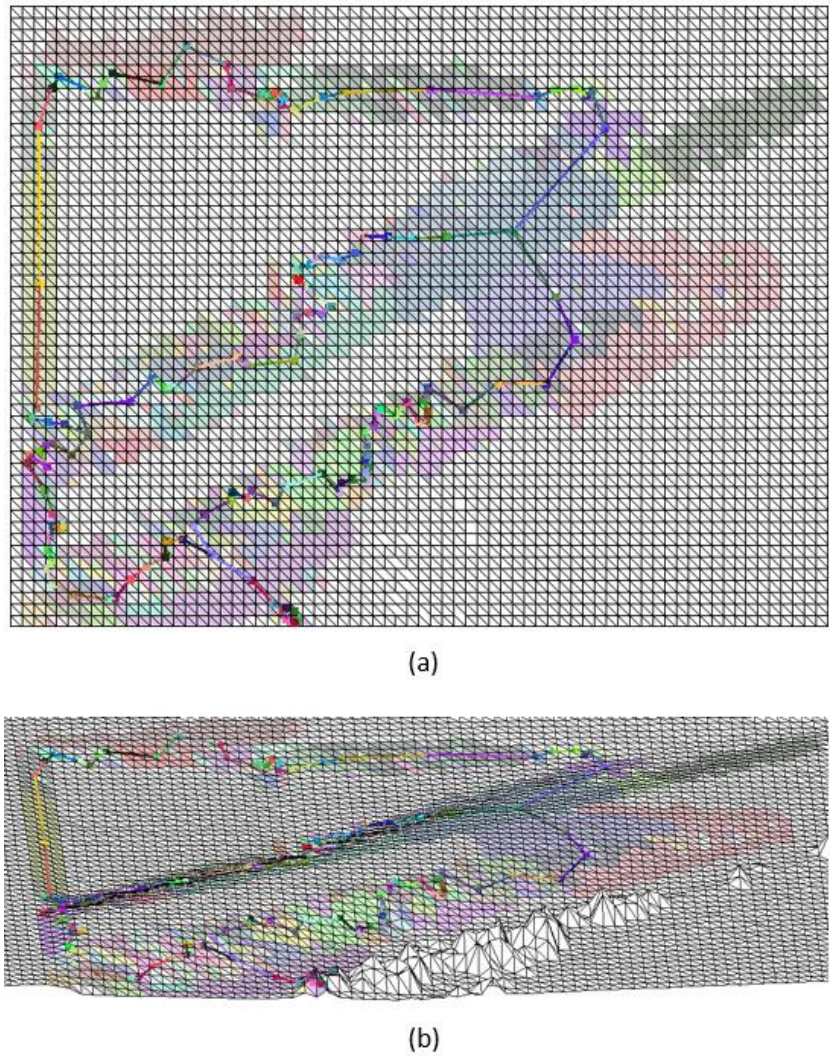


Figure 65 Illustration displaying representative_path_loop of hill shaped and watershed shaped landform at (a) top view, and (b) oblique view

7.3 Extracting object_path_loop

The representative_path_loops extracted in the previous step indicates the presence of object and sub-objects in the scene. However, the loops do not map out the boundary of the object. Since the perimeter of an object constitutes planar segments that belong to the foothill_slope and thereby can be represented by a path_loop (Figure 15). This special path_loop is termed as object_path_loop. A sub-algorithm is designated to determine the object_path_loop of the object and sub-object using the representative_path_loop and the complete_flow_graph of the object segment. Figure 66 shows the flowchart of the sub-algorithm employed to extract the object_path_loops.

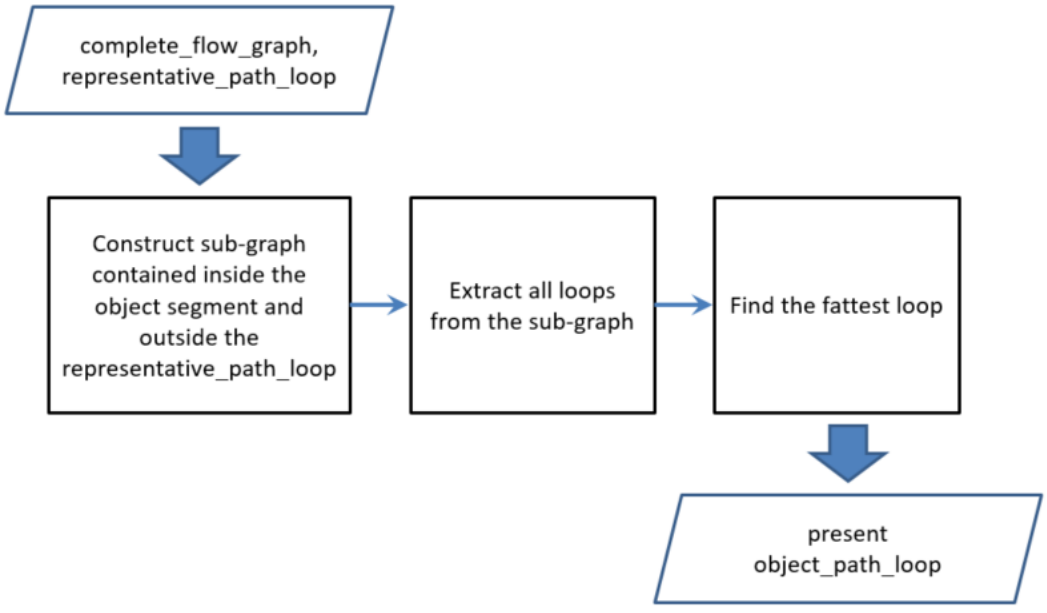


Figure 66 Flowchart of the sub-algorithm that extract object_path_loop

An example of the complete_flow_graph generated from the naïve segment (Figure 41) is shown in Figure 67. The corresponding representative_path_loop is shown in Figure 68.

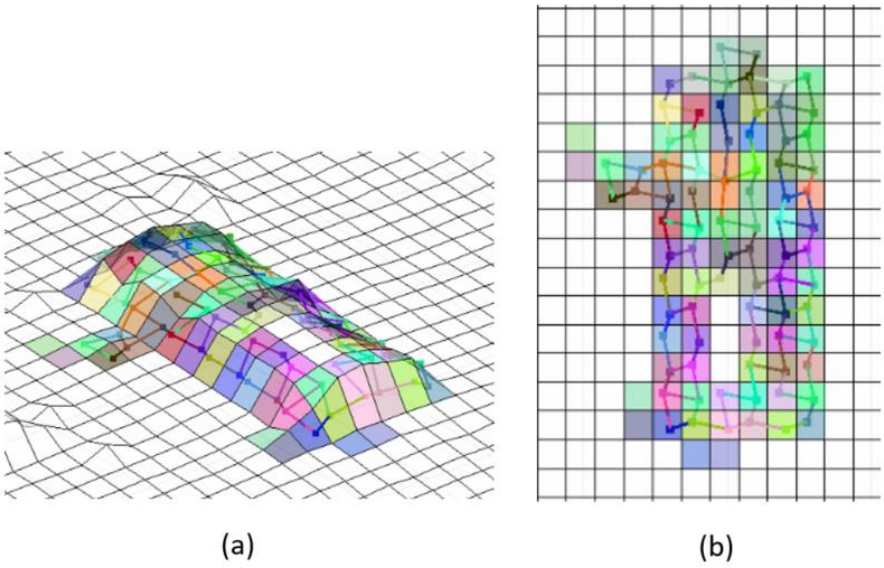


Figure 67 Complete_flow_graph from object segment shown in Figure 41 (a) oblique view (b) top view

The algorithm first extracts the sub-graph present inside the representative_path_loop. The sub-graph is then subtracted from the complete_flow_graph. The sub-graph that remains is shown in Figure 69. Since the representative_path_loop uniquely represent the corresponding object, graph representing any sub-object get excluded within the extracted sub-graph. Thereby, path_loops that can be extracted from the sub-graph belongs to the targeted object.

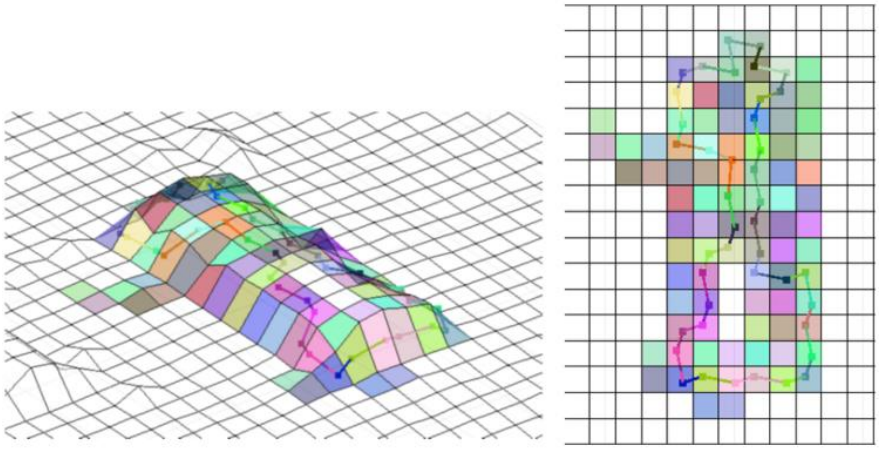


Figure 68 Representative_path_loop of the object (a) oblique view (b) top view

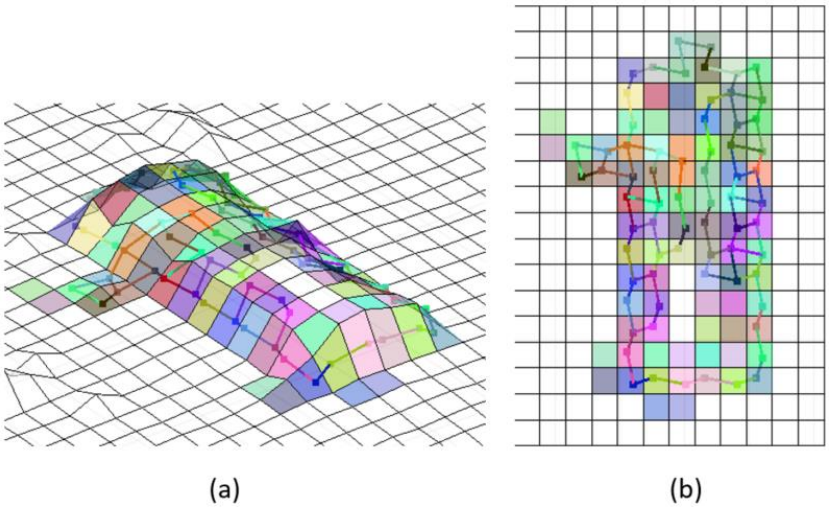


Figure 69 Sub-graph of the object (a) oblique view (b) top view

All the path_loops present in the sub-graph is extracted by the method described in Section 7.2. Next, the algorithm finds out the path_loop that contains

within its boundary the maximum number of nodes. That path_loop is our desired object_path_loop. An example of the object_path_loop is shown in Figure 70.

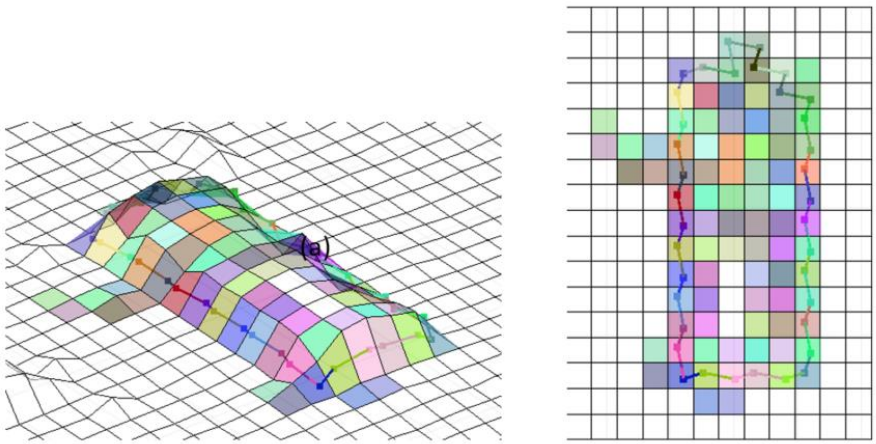


Figure 70 Object_path_loop of the object (a) Oblique view (b) top view

The object segment is thereby defined by the group of planar segments within the confines of the object_path_loop. Figure 71 shows the extracted object segment of the example. Comparing the result with the naïve segment (Figure 41), It is clearly evident that most of the false-positive has been removed due to this process.

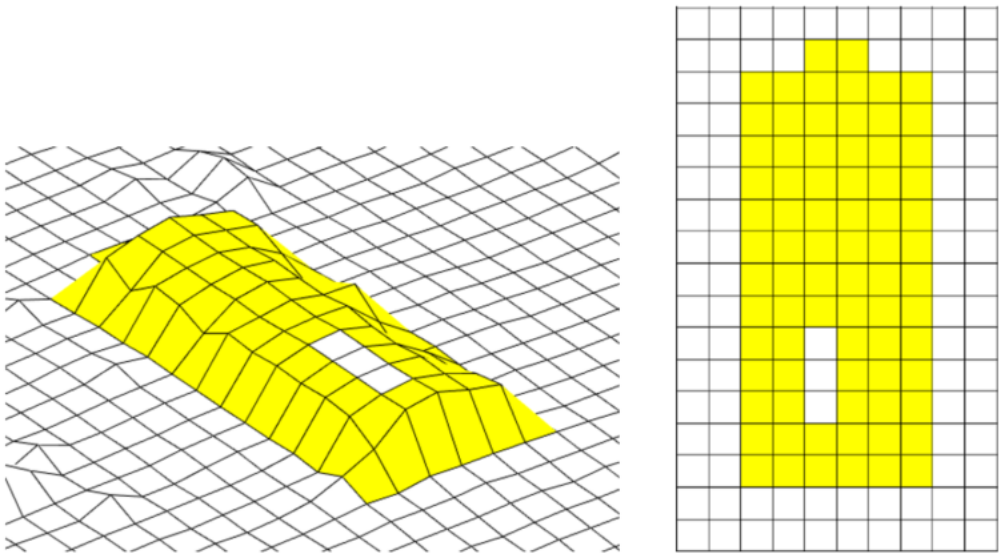


Figure 71 Extracted object after using Hierarchical decomposition on object segment shown in Figure 41

7.4 Objectness test

Generation of representative_path_loop from the complete_flow_graph is based on the principle of object geometric property such as concavity or convexity. This may cause some false positive results as any non-object which have concave shape structure may produce a representative_path_loop. Even a slightly elevated corner of a rectangular object can be extracted as sub-object in this process. That is why every extracted object segments are subjected to a validity test which filter out the non-objects. We call this test '*Objectness test'* and is based on a quantity called *'objectness'*. Figure 72 shows a simulated example of an object and its sub-object.

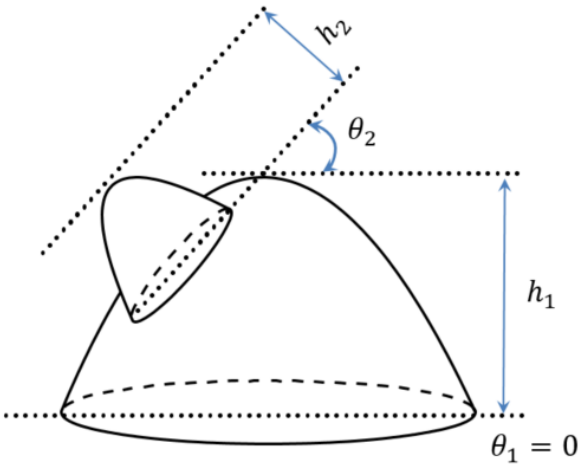


Figure 72 A simulated example of an Object and its sub-object

The objectness is calculated using two measures: the maximum height of the object from its base (*h*) and the angle between the object normal and the BG normal (θ). That is why, in *Figure 72* the measure of object 1 is h_1 and θ_1 (which is 0) and the measure of object 2 is h_2 and θ_2. If the height and the angle is above a certain threshold value, the segment is classified as object. Otherwise, the structure is rejected as non-object.

7.5 Find Object type

Next, the path loops extracted from the previous stages are used to classify the object as belonging to either of two groups: convex-like object and concave-like object. A procedure is dedicated to serve this purpose.

As discussed in section 4.1, in the case of a convex-like object, the surveyor motion is in counter-clockwise direction whereas, for the concave-like object, the motion is clockwise. The exterior angle that corresponds to each connecting edge forming the loop is determined. A rule of the polygons is that the sum of the exterior angles always equals 360 degrees. Therefore, in case of concave-like object as shown in Figure 73(a), sum of exterior angle = $\sum_i \theta_i$ = -360°. In case of convex-like object (Figure 73 (b)), sum of exterior angle = $\sum_i \theta_i$ = 360°. The sign of the sum hints the class of the object.

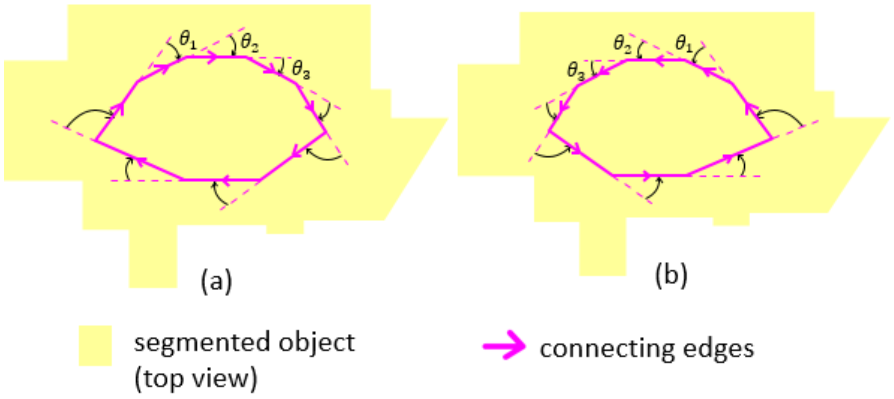


Figure 73 Simulated example of a path loop around an (a) horizontal convex structure (b) horizontal concave structure

7.6 Associate planar segments to representative_path_loops

In the previous steps, the parent object has been delineated from its topographic background. The representative_path_loop of the sub-object reveals its presence and position relative to the parent object. The next step is to find the boundaries of the sub-objects. In the case of the parent object, the boundary was determined based on the sub-graph extracted using the naive segment (which provide the upper-limit of the boundary) and the representative_path_loop (which provide the lower-limit of the boundary). However, in the case of sub-object, there is no such segment available that specifically includes the foothill_slope individual sub-object (referred as 'sub-object region'). A sub-algorithm is assigned to estimate the sub-object region. In the next two subsequent sub-section, the

underlying mechanism for this method is discussed and the flowchart of the algorithm is shown and discussed.

7.6.1 Underlying mechanism

In section 7.1 we arrive at the conclusion that, each object in a topographic map can be represented by a set of circular path loops where the dot product between their Path vector and the Surveyor Guidance vector at each point has the same sign i.e. the loop satisfies the path_loop condition. One such circular path loop termed as representative_path_loop is considered as shown in Figure 74(a).

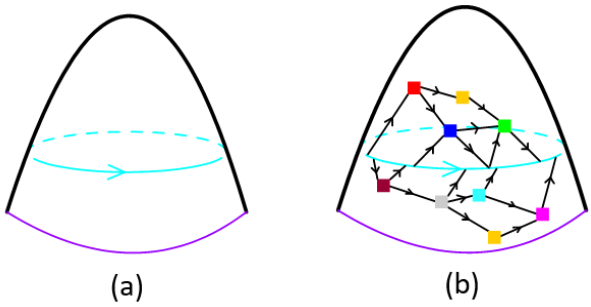


Figure 74 Convex structure object with (a) closed path loop and (b) the points connected to the loop

Since a path_loop can be associated with each point of the object, and the representative_path_loop is formed by the merge of all such path_loops. So, it can be postulated that every point on the object can be connected to the loop via a

chain of connecting edges that satisfy the path_loop condition. A stylistically simulated example is shown in Figure 74 (b). This means it is possible to associate each planar segment of the object to the representative_path_loops with a chain of connecting edges.

A stylistically simulated example of a complete_flow_graph of a complex object segment is shown in Figure 75.

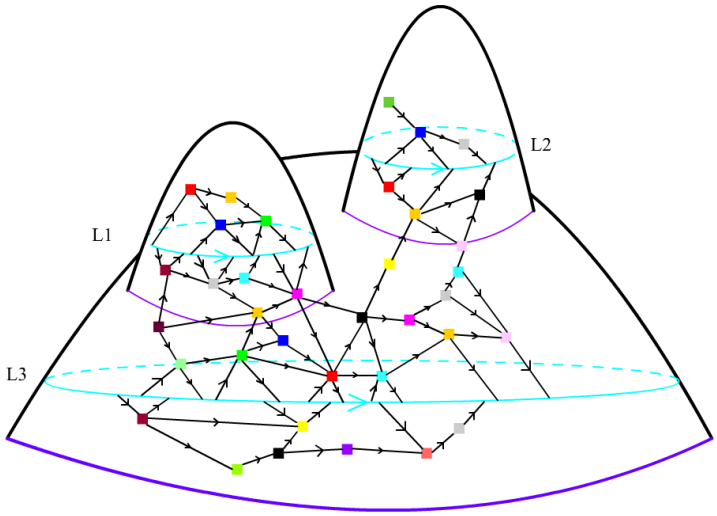


Figure 75 Illustration displaying the complete_flow_graph of a composite object segment with representative_path_loops in cyan color

Since a number of nodes at the edge of the sub-objects is connected to a number of nodes belonging to the parent object and since each node is connected to the corresponding representative_path_loops, it can be postulated that the

representative_path_loop of the sub-objects is connected to the representative_path_loop of the parent object via a chain of connecting edges.

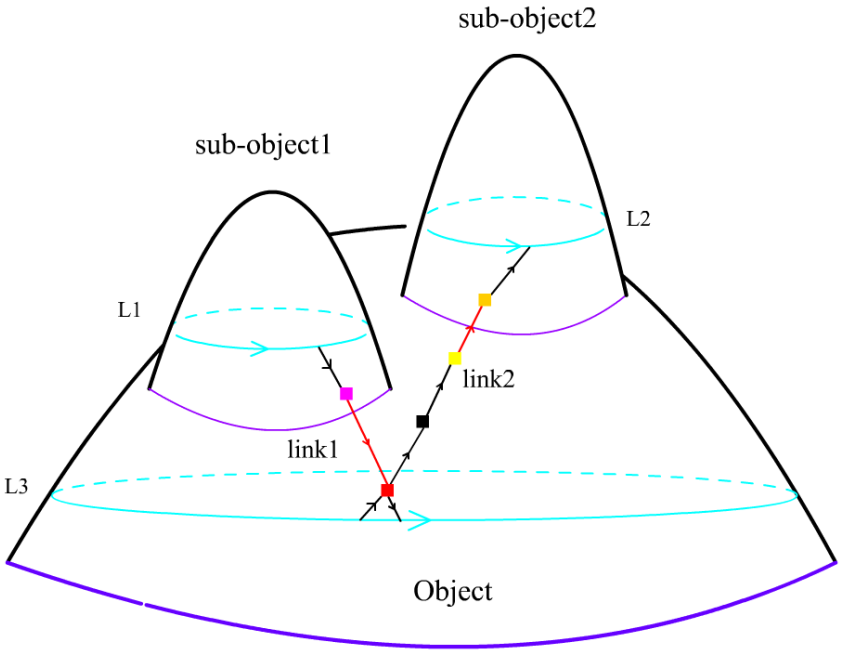


Figure 76 Representative_path_loop of every sub-object is connected to the representative_path_loop of the object via a chain of connecting edges

Here the nodes are represented by small colored squares. Representative_path_loop of the sub-object L1 is connected to the representative_path_loop of the parent object L3 via two nodes whereas L2 is connected to the L3 via 4 nodes.

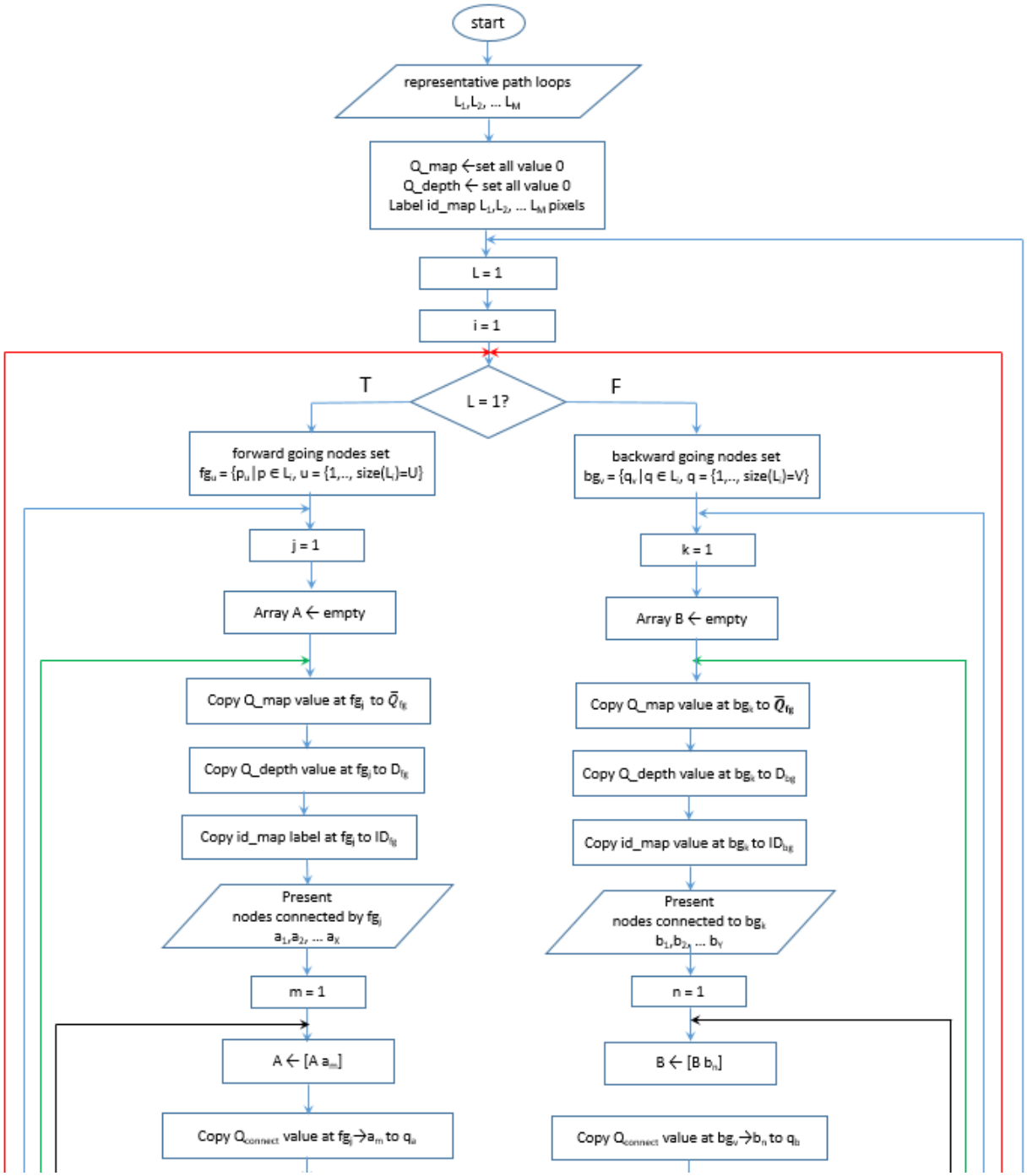
One of the underlying assumptions is that a sub-object is bound by topographic discontinuities. This means the connecting edge that crosses over that boundary (connection colored red in Figure 76) are weak and therefore the

$Q_{connection}$ value of that connection would be low. This means, Q average value ($\bar{Q}$) of the connecting edges that take that route to the parent representative_path_loop L3 would be lower than those routes to the local representative_path loop (L1 or L2). Therefore, to associate the planar segments (nodes) to its corresponding representative_path_loop, the following strategy can be adopted.

For each node in the composite object, the minimum Q average value ($\bar{Q}$) of the chain of connecting edges to each of the representative_path_loops (if there exist a connection) is measured. Among the competed representative_path_loops, the one that gives the lowest Q average value ($\bar{Q}$) is selected as the corresponding representative_path_loop

7.6.2 Flow chart

Figure 77 shows the detailed flow chart of the algorithm that is responsible for the planar segments association to the representative_path_loops.


start
representative path loops
$L_1, L_2, \ldots L_M$
Q_map ←set all value 0
Q_depth ← set all value 0
Label id_map $L_1, L_2, \ldots L_M$ pixels
L = 1
i = 1
T
L = 1?
F
forward going nodes set
$fg_u = \{p_u \mid p \in L_i, u = \{1,.., size(L_i)=U\}$
backward going nodes set
$bg_v = \{q_v \mid q \in L_i, q = \{1,.., size(L_i)=V\}$
j = 1
k = 1
Array A ← empty
Array B ← empty
Copy Q_map value at fg_j to $\bar{Q}_{fg}$
Copy Q_map value at bg_k to $\bar{Q}_{fg}$
Copy Q_depth value at fg_j to D_{fg}
Copy Q_depth value at bg_k to D_{bg}
Copy id_map label at fg_j to ID_{fg}
Copy id_map value at bg_k to ID_{bg}
Present
nodes connected by fg_j
$a_1, a_2, \ldots a_X$
Present
nodes connected to bg_k
$b_1, b_2, \ldots b_Y$
m = 1
n = 1
A ← [A a_m]
B ← [B b_n]
Copy $Q_{connect}$ value at $fg_j \rightarrow a_m$ to q_u
Copy $Q_{connect}$ value at $bg_v \rightarrow b_n$ to q_b

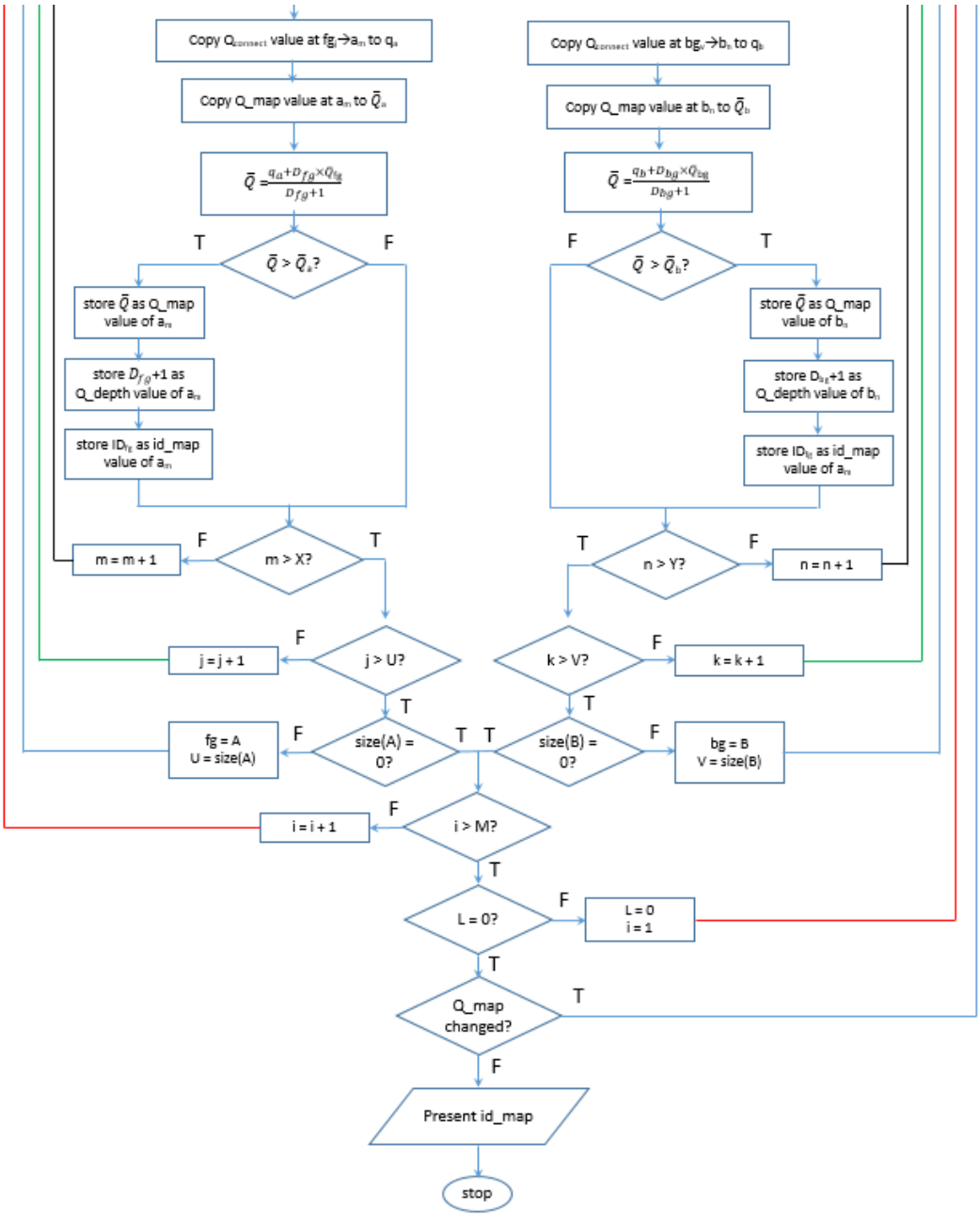


Figure 77 Flowchart of the algorithm that associates planar segments to the representative_path_loops

The procedure begins with labeling the representative_path_loops in a map referred to as 'id_map'. The procedure also maintains two other maps called 'Q_map' and 'Q_depth' throughout the process. They are initialized to zero. The objective of the procedure is to identify objects and sub objects membership for each node of the considered object segment i.e. to fill the id_map up with the labels. This is achieved by associating each planar segment to a representative_path_loop such that the overall Q average value ($\bar{Q}$) measured for each planar segments in the object segment from their connection to its corresponding representative_path_loop is the maximum. At the end of the run, each planar segment in the Q_map would store the optimized Q average value ($\bar{Q}$) and in case of Q_depth, each planar segment would store the distance, in terms of number of connecting edges forming the connection, to the corresponding representative_path_loop. In the procedure, at each iteration, the planar segments in the object segment are explored gradually via the complete_flow_graph and the Q_map and Q_depth map are updated. For a given planar segment, Q average value ($\bar{Q}$) is calculated for each of its neighbors based on its own Q average value, Q_{fg} (extracted from the Q_map) and the depth value, D_{fg} (extracted from the Q_depth) and also the $Q_{connect}$ value of the connection, q_a (extracted from the $Q_{connect}$_map). The formula used for Q average value ($\bar{Q}$) estimation is given below:

$$\bar{Q} = \frac{q_a + D_{fg} \times Q_{fg}}{D_{fg} + 1} \quad \textit{eq. 12}$$

If the Q average value ($\bar{Q}$) estimated is greater than the Q_map value of the neighboring planar segment, that value is replaced by Q average value ($\bar{Q}$) and

both the Q_depth map and the id_map are updated accordingly. The whole procedure is run several times until no apparent change is detected in the Q_map with each cycle. This ensures that the Q_map has reach its globally optimum state. Appendix D demonstrates how this optimization ensures proper distribution of labels of planar segments in the object segment.

7.7 Hierarchical Decomposition result

Figure 78 shows a complex landforms segment (Figure 42) decomposed to its constituent landforms. Their neighborhood and hierarchical relationship are also specified in the figure.

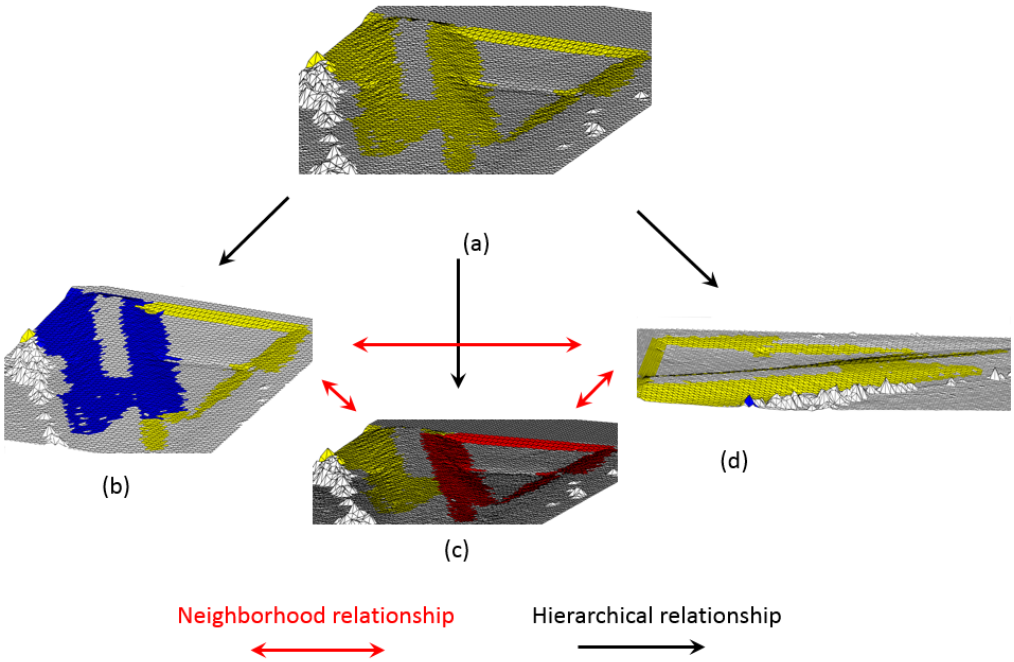


Figure 78 (a) Landform segmentation decompose to (b & d) two horizontal convex structure and (c) one concave structure. See also Figure 42

8. Performance Analysis

In this chapter, the performance of the Virtual Surveyor based object extraction method is discussed. Since segmentation of the parent object and its sub-objects is carried out in a recursive pattern, in the performance analysis, we focus mainly on the segmentation of the parent object. In the first subsection of the chapter, the qualitative result of the algorithm is examined and in the subsequent subsection, result from the quantitative analysis is shown.

8.1 Qualitative Analysis

In this section, we discussed some of the qualitative results that demonstrate the efficacy of the algorithm. Here, two man-made structures are considered as shown in Figure 79. Figure 79 (a) is a complex building structure, whereas Figure 79 (b) is a pair of adjacent pools. The latter is a convex structure, whereas the former is a concave structure. Figure 80 shows the corresponding results obtained by applying the Pfeifer's ground filter. The region colored red is labeled as ground.

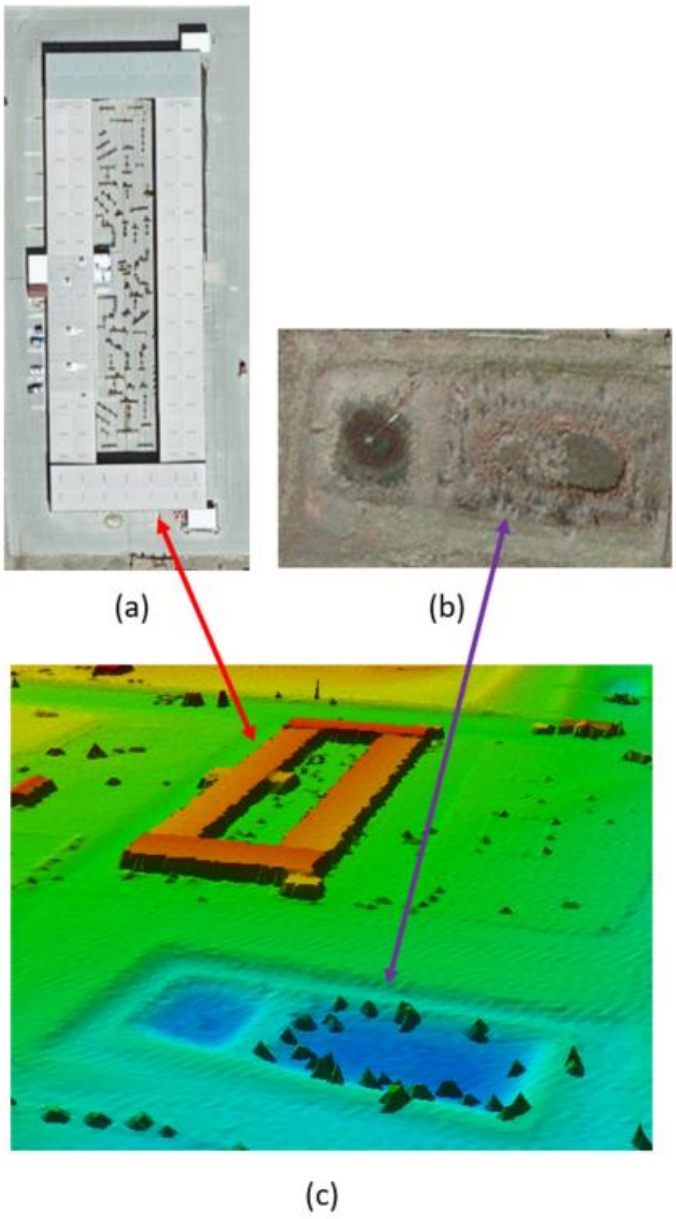
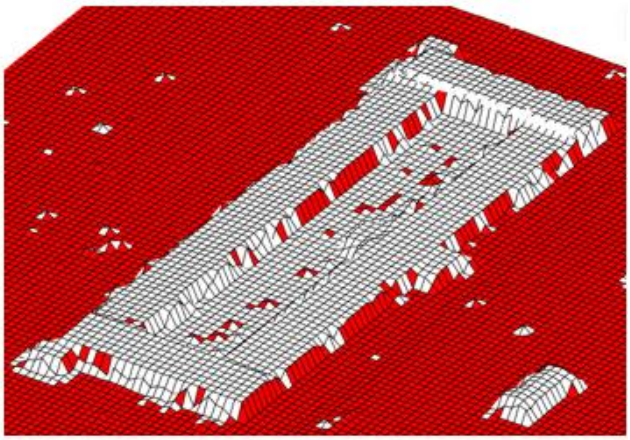
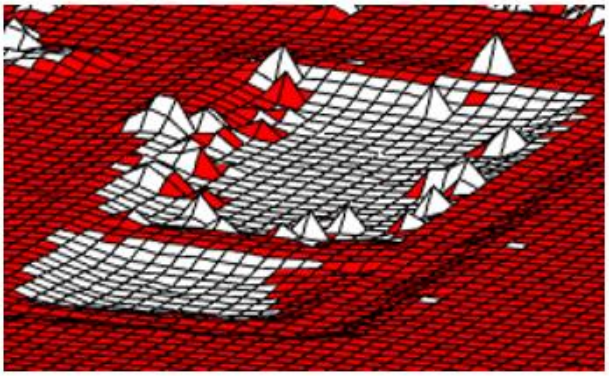


Figure 79 Top-view RGB image of (a) complex building (b) pools and (c) elevation image



(a)



(b)

Figure 80 Pfeifer's ground filter results. Red planar segments - ground pixel, White planar segments – object pixel

As evident from the figure (Figure 80), the object image is contaminated with salt and pepper noise, which is a common trait for the pixel-based ground filter. The courtyard of the building is labeled as non-ground. This is due to the fact that the ground filter has been modified to detect man-made cavities on the ground in addition to man-made convex structure. For some user, this might be undesirable as the courtyard is usually regarded as ground. In Figure 80 (b), it is

evident that the bank of the pool, which is part of the man-made structure, has not been labeled as an object class. This interpolation based filter underperforms at the vicinity of a low-grade slope's edge. Figure 81 shows the corresponding results obtained by applying the Virtual Surveyor based object extraction algorithm. As can be seen from the figure, sub-objects that lie on the foothill_slope of the parent objects are also included in the resultant segment.

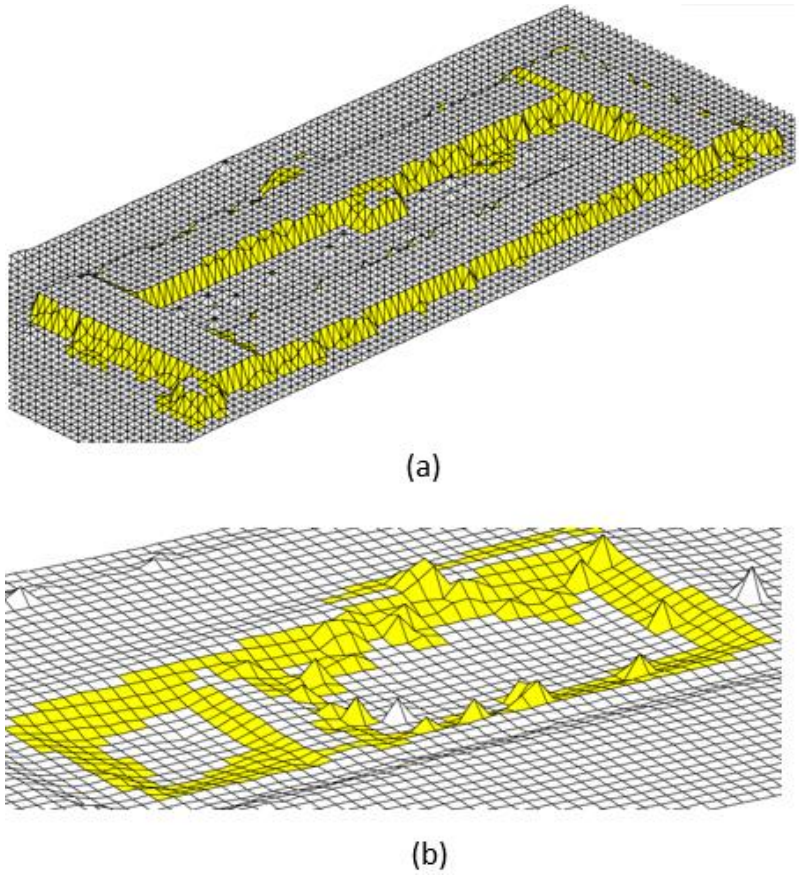


Figure 81 Virtual Surveyor Segmentation result

Figure 82 shows the hierarchical decomposition of the extracted image object of the complex building. There is multi-level decomposition obtained at a single scale.

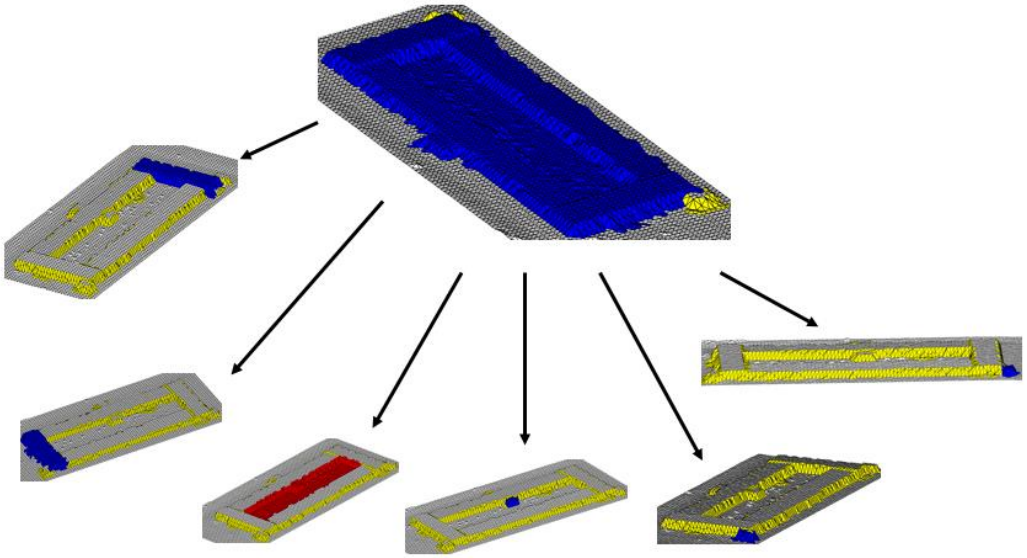


Figure 82 Hierarchical decomposition of the complex building. Blue segments represent convex structure whereas red segments represent concave structure.

In the first level, the image object is first decomposed into three neighboring objects: the main building and the two small rectangular block adjacent to the corner at the opposite side of the main building. The building is then further decomposed into its major components. The courtyard is clearly identified and segmented out. Here, the blue represents a convex object, whereas, the red

represents concave objects. It is to be noted that there are two external objects at the side of the building that was included during the segmentation process has also been extracted as external objects.

Figure 83 shows the hierarchical decomposition of the extracted image object of the adjacent pair of pools.

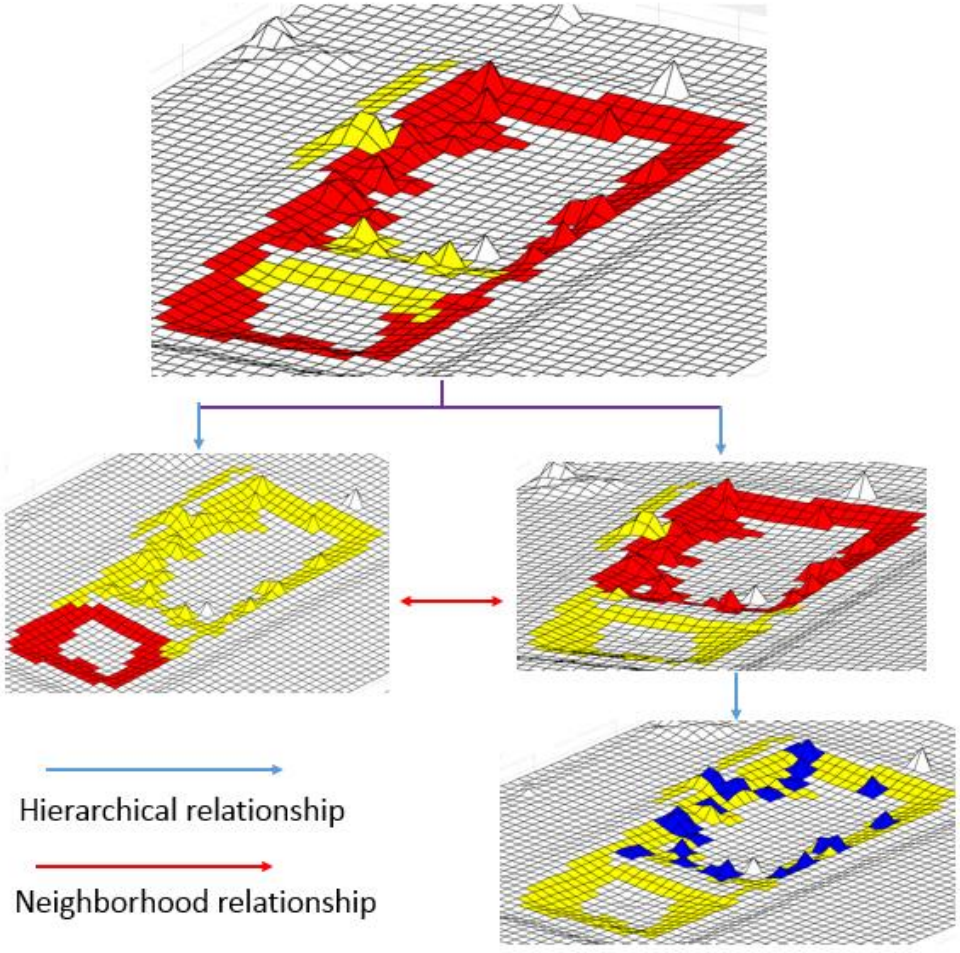


Figure 83 Hierarchical decomposition of the pair of pools

Here, too, multi-level decomposition is conducted without traversing through multiple scales. In the first level, the pools are separately identified and segmented out. One of the pools is further decomposed into its sub-objects. The individual trees are also revealed in the result as shown in the figure.

The VSOE algorithm is able to separate objects that lie in close proximity. An example is of two high proximity neighboring objects is shown in Figure 84 (a) and (b). Figure 84 (c) and (d) shows their representative_path_loops from different views.

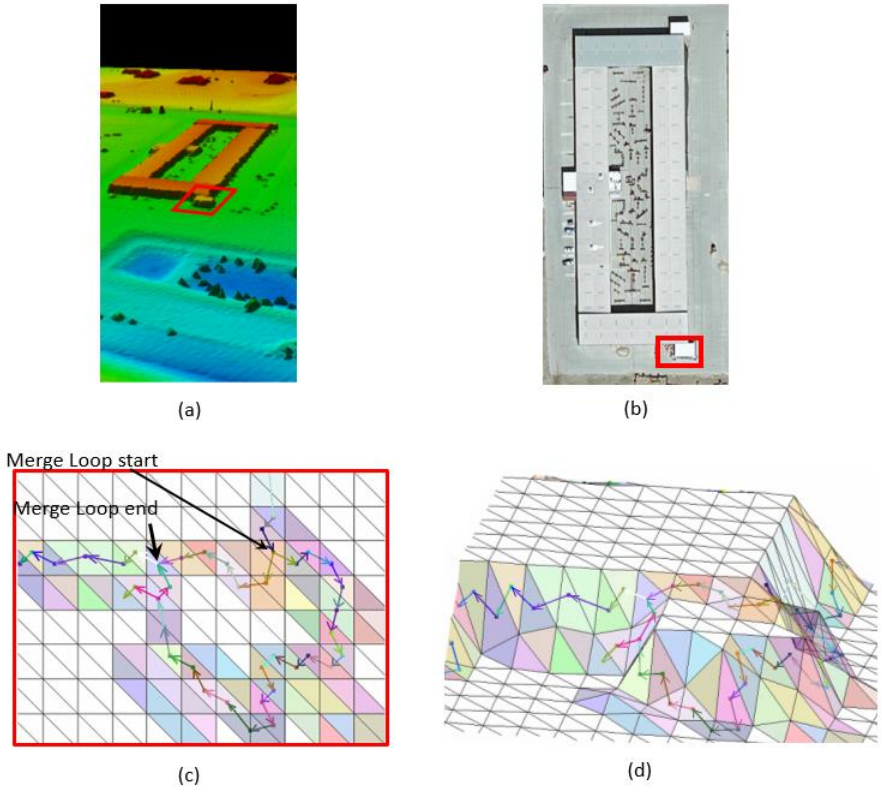


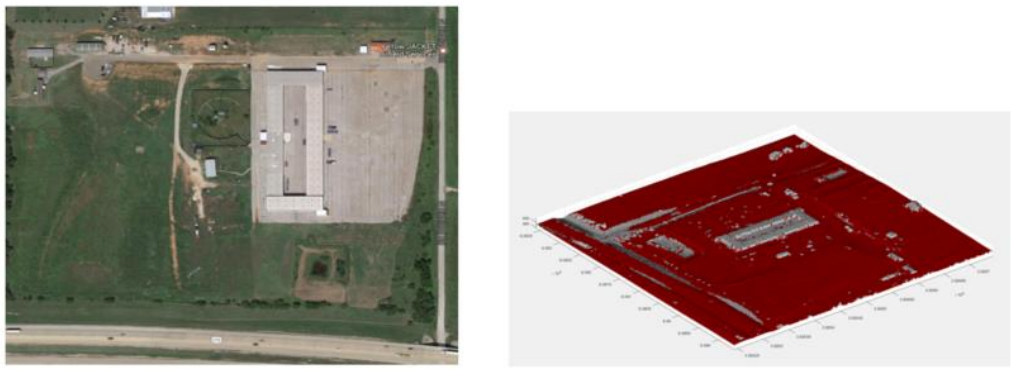
Figure 84 Illustration displaying a (a) 3D point cloud model of the complex building adjacent to a small rectangular object (highlighted by a red bounding

box), a (b) orthophoto of the complex building, (c) close up top view of the neighboring rectangular object superimposed by their representative_path_loops, (d) oblique view

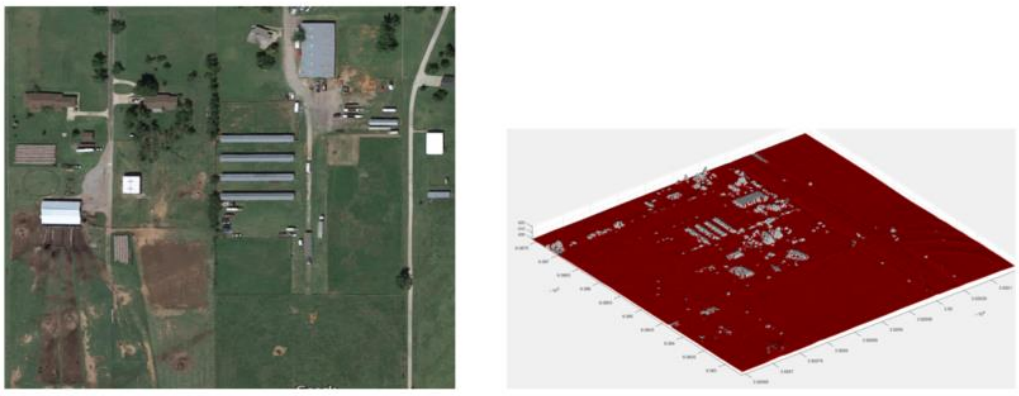
Section 7.2.3 explains the formation of representative_path_loops for both objects. They, therefore, can be easily separated during the Hierarchical decomposition step. It is evident from the Hierarchical decomposition result of the complex building in figure 85 that the neighboring objects have been cleanly separated.

8.2 Quantitative Analysis

In this section, the quantitative analysis of the algorithm performance is discussed. Here, we compare the performance of object extraction by Pfeifer's ground filter and that by VSOE. This will demonstrate the effectiveness of VSOE to improve upon the result of the Pfeiffer's ground filter. To test VSOE, we build a database of 78 small object candidates collected after applying Pfeifer's ground filter on two different terrains shown in Figure 86.



(a)



(b)

Figure 86 Object candidates collected from these two terrains.

In the figure, on the right-hand side, the top view of the RGB image of the terrain is shown whereas, on the left-hand side, the result obtained by the application of Pfeifer's ground filter on these terrains are shown. Here, the red planar segments are the ground pixel and the white planar segments are the object pixels. A sample of the object candidates that are collected from the terrains is shown in Figure 87.

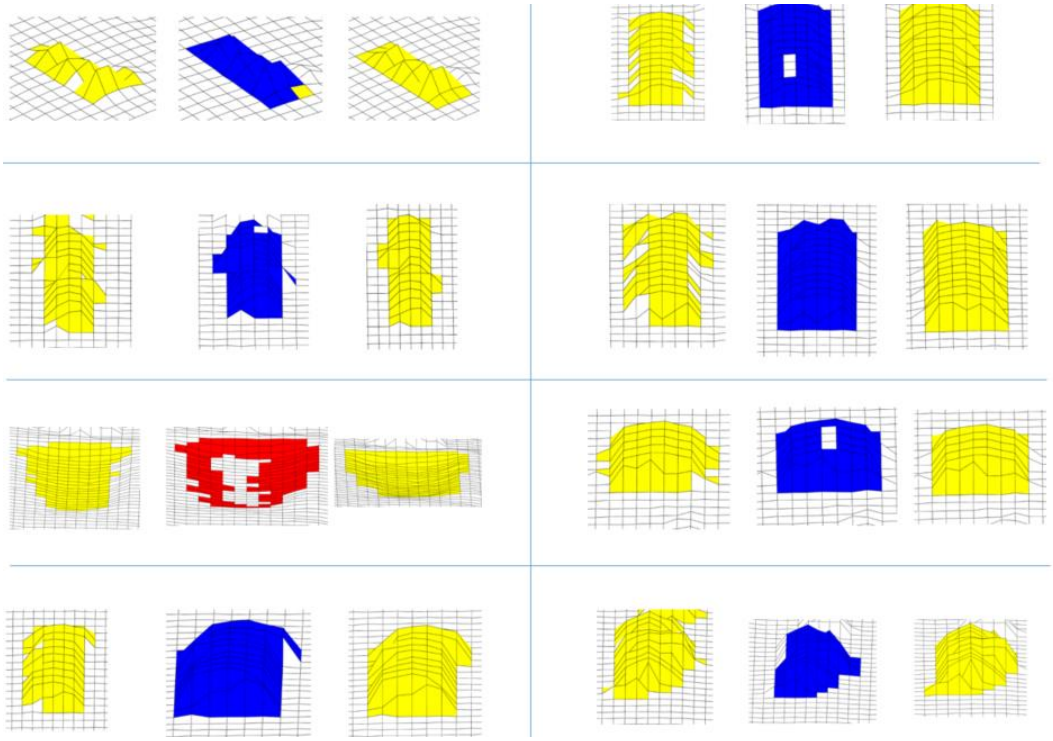


Figure 87 Eight samples of the given test objects, the first column represents Pfeiffer's ground filter result, the second column represents VSOE result, the third column represents the ground truth

The ground truth of the segmentation for each sample is shown in the third column. The ground truth is obtained manually based on human judgment. In the database, we only consider small objects since manually drawing the boundaries on the large object is time-consuming and error prone. Large objects such as shown in Figure 79 are ignored.

The measurement metric adopted for the performance comparison between the filter results and the ground truth is based on the number of common square elements that can be found in the raster representations. Since, in the performance analysis, the boundary of the foothill_slope is what matters, so we only consider the square elements that form the boundary of the object segment.

A segmentation result is usually evaluated by assessing its consistency with the ground truth segmentation (Van Rijsbergen, 1979). Here we use the F_1 -score for consistency. F_1-score is based on the precision and recall values of the segmentation result. Precision and recall is defined as

$$P = \frac{TP}{TP+FP} \qquad \text{eq. 13}$$

$$R = \frac{TP}{TP+FN} \qquad \text{eq. 14}$$

Where, TP = True Positive, FP = False Positive and FN = False Negative. F_1-score is defined as

$$\mathrm{F}_1 - score = 2\frac{R \times P}{P+R} \qquad \text{eq. 15}$$

The precision, recall and the F_1 – score is shown in Table 3. The table clearly demonstrates the efficacy of the AVSOE algorithm

Table 3 Segment coverage test result

	Precision	Recall	F_1 – score
Pfeifer's GF	0.933	0.83	0.878
AVSOE	0.989	0.968	0.978

9. Conclusion and Suggested Future Work

In this work, we introduced a fully automatic algorithm, called Automatic Virtual Surveyor based Object Extraction (AVSOE) that works on the airborne LiDAR point cloud data of a given terrain to segment out natural as well as man-made objects. The algorithm is designed to tackle two main challenges facing object based image analysis: scale dependency and the object complexity. The 3D object segmentation method is based on the premise that an object, no matter how complex, introduces a distinct concavity or convexity onto its topographic background. The algorithm starts by constructing a directed graph from the mesh representation of the point cloud data of the terrain with nodes representing the cell of the mesh and the edges traverse perpendicularly to the direction of the sloping surface. As a result, each object and sub-objects present in the scene are encircled by cycle graphs. Detection of these cycle graph and subsequent processing reveals simultaneously the presence of the objects and sub-objects, their positions, their hierarchical relations and their associated voxels. Thereby, objects of all size and shape are extracted in a single scale level.

The non-parametric and scale independent algorithm is therefore capable of automatically segmenting, at a single scale level, a complex object such as forest and also each of the constituent sub-objects such as the individual tree composing the forest. The proposed algorithm is therefore non-parametric and scales independently.

9.1 Contribution

In this research, the key contributions are:

1. In this work, we propose a method, which suggests a paradigm shift in the way 3D object segmentation is approached. The method provides a generic solution to extract 3D objects

2. Existing 3D object extraction method only segment convex objects such as trees, building etc. Our method can extract not only convex structured objects but also concave structured objects.

3. In most object extraction methods, a compromise has to be made between Type-I error and Type-II error. However, using this method, Type-I error and type-II error can both be minimized.

4. Existing 3D object segmentation method mostly rely on the distinctive geometric properties of the object, such as the proximity between neighboring LiDAR points. In some cases, (mostly natural objects), it is not clear, where the object ends and the background begins. Moreover, it can prove difficult for the method to distinguish objects that lie in close proximity.

Our method is designed to extract objects overlaying the topographic surface irrespective of the grade of its slope or its proximity to other objects in the scene.

5. Due to the relative complexity of real-world objects and the difficulty in setting scale parameters tuned to each class of objects present in the scene, over-

segmentation is an issue in the results of existing object extraction method. The problem is further exacerbated in the case of high-quality data since the heterogeneity present in the image-object become more pronounced. Since segmentation quality has a direct impact on object-based classification result, the performance of the traditional methods is below par. In addition, due to the imprecise selection of the scale parameter, boundaries do not perfectly overlap (coincide) through scale. Therefore, hierarchical relation is difficult to establish among object and its sub-objects.

Our proposed method is independent of scale parameters and is designed to segment convex and concave structure representing the image object regardless of the extent of internal heterogeneity present in the structure. The boundaries shared by the object and its sub-objects emerge simultaneously in the hierarchical decomposition process, and the hierarchical relations are explicitly expressed.

9.2 Future work

As a future work, the following things could be attempted:

1. AVSOE automatically segments the 3D object and its sub-objects. The next step is to identify the object. As discussed in section 3.2, 3D object segmentation enables the classifier to exploit a range of new features such as aggregated spectral pixel values, shape, texture, context as well as topology.

2. The data set delivered by the new LiDAR sensor technology such as the Geiger-mode and the single photon is much richer than the traditional linear mode. Due to the 360-degree view of the new sensor, it generates points of high density on the vertical features of the object. Our object extraction method is designed to leverage this newly revealed aspect of the object. It would be interesting to see the performance of AVSOE on those data set.

References

P. Acharjee,G. Toscano,C. McCormick and V. Devarajan, “Performance analysis of a novel algorithm for large scale water-body surface mapping using elevation and intensity of LiDAR data,” Imaging and Geospatial Technology Forum, IGTF 2016 - ASPRS Annual Conference and co-located JACIE Workshop. American Society for Photogrammetry and Remote Sensing, Fort Worth, Texas, 2016

A. Antonarakis, K. S. Richards, and J. Brasington, “Object-based land cover classification using airborne LiDAR,” Remote Sensing of Environment, vol. 112, no. 6, pp. 2988–2998, 2008.

H. Arefi and M. Hahn, “A hierarchical procedure for segmentation and classification of airborne LIDAR images,” in INTERNATIONAL GEOSCIENCE AND REMOTE SENSING SYMPOSIUM, 2005, vol. 7, p. 4950.

P. Axelsson, “DEM generation from laser scanner data using adaptive TIN models,” International Archives of Photogrammetry and Remote Sensing, vol. 33, no. B4/1; PART 4, pp. 111–118, 2000.

M. Baatz and A. Schäpe, “Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation,” Angewandte Geographische Informationsverarbeitung XII, vol. 58, pp. 12–23, 2000.

T. Blaschke, “Object based image analysis for remote sensing,” ISPRS journal of photogrammetry and remote sensing, vol. 65, no. 1, pp. 2–16, 2010.

T. Blaschke et al., “Geographic object-based image analysis–towards a new paradigm,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 87, pp. 180–191, 2014.

N. Brodu and D. Lague, “3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 68, pp. 121–134, 2012.

Y. Cui, L. Chapel, and S. Lefèvre, “Combining multiscale features for classification of hyperspectral images: a sequence based kernel approach,” in Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2016.

Z. Chen, B. Devereux, B. Gao, and G. Amable, “Upward-fusion urban DTM generating method using airborne Lidar data,” ISPRS journal of photogrammetry and remote sensing, vol. 72, pp. 121–130, 2012.

C. Chen, Y. Li, W. Li, and H. Dai, “A multiresolution hierarchical classification algorithm for filtering airborne LiDAR data,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 82, pp. 1–9, 2013.

Chen, Z.Y.; Gao, B.B. An Object-Based Method for Urban Land Cover Classification Using Airborne Lidar Data. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2014, 10, 4243–4354.

Q. Chen, H. Wang, H. Zhang, M. Sun, and X. Liu, “A point cloud filtering approach to generating DTMs for steep mountainous areas and adjacent residential areas,” Remote Sensing, vol. 8, no. 1, p. 71, 2016.

Z. Chen, B. Xu, and B. Gao, “An Image-Segmentation-Based Urban DTM Generation Method Using Airborne Lidar Data,” IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 9, no. 1, pp. 496–506, 2016.

G. Cheng and J. Han, “A survey on object detection in optical remote sensing images,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 117, pp. 11–28, 2016

D. Costantino and M. G. Angelini, “Features and ground automatic extraction from airborne LiDAR data,” Int Arch Photogramm Remote Sens Spat Inf Sci, vol. 38, no. 5/W12, 2011.

F. Crosilla, D. Macorig, M. Scaioni, I. Sebastianutti, and D. Visintini, “LiDAR data filtering and classification by skewness and kurtosis iterative analysis of multiple point cloud data categories,” Applied Geomatics, vol. 5, no. 3, pp. 225–240, 2013.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE transactions on evolutionary computation, vol. 6, no. 2, pp. 182–197, 2002.

S. D’Oleire-Oltmanns, C. Eisank, L. Drăguţ, L. Schrott, I. Marzolff, and T. Blaschke, “Object-Based Landform Mapping at Multiple Scales from Digital Elevation Models (DEMs) and Aerial Photographs,” Proceedings of the 4th GEOBIA, pp. 496–502, 2012.

P. Domingos, “A few useful things to know about machine learning,” Communications of the ACM, vol. 55, no. 10, pp. 78–87, 2012.

L. Drăguţ, O. Csillik, J. Minár, and I. S. Evans, “Land-surface segmentation to delineate elementary forms from Digital Elevation Models,” 2013.

L. Drăguţ, D. Tiede, and S. R. Levick, “ESP: a tool to estimate scale parameter for multiresolution image segmentation of remotely sensed data,” International Journal of Geographical Information Science, vol. 24, no. 6, pp. 859–871, 2010.

D. C. Duro, S. E. Franklin, and M. G. Dubé, “A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using SPOT-5 HRG imagery,” Remote Sensing of Environment, vol. 118, pp. 259–272, 2012.

I. S. Evans, “Geomorphometry and landform mapping: What is a landform?,” Geomorphology, vol. 137, no. 1, pp. 94–106, 2012.

I. S. Evans and J. Minár, “A classification of geomorphometric variables,” in Proceedings of the Geomorphometry 2011 Conference, Redlands, California, USA, 2011, pp. 105–108.

A. Golovinskiy, V. G. Kim, and T. Funkhouser, “Shape-based recognition of 3D point clouds in urban environments,” in 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 2154–2161.

B. Guo, X. Huang, F. Zhang, and G. Sohn, “Classification of airborne laser scanning data using JointBoost,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 100, pp. 71–83, 2015.

G. J. Hay and G. Castilla, “Geographic Object-Based Image Analysis (GEOBIA): A new name for a new discipline,” in Object-based image analysis, Springer, 2008, pp. 75–89.

Y. Hu, Automated extraction of digital terrain models, roads and buildings using airborne LiDAR data, vol. 69. 2004.

C.-M. Huang and Y.-H. Tseng, “Plane Fitting Methods of Lidar Point Cloud,” in In 29th Asian Conference on Remote Sensing (ACRS), Beijing, November 2008, pp. 1040 – 1044

L. Hui-ying, X. Yu-jun, W. Zhi, and L. Yi-nan, “Hierarchical Algorithm in DTM Generation and Automatic Extraction of Road from LIDAR Data,” ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 1, pp. 133–136, 2012.

B. Koch, U. Heyder, and H. Weinacker, “Detection of individual tree crowns in airborne lidar data,” Photogrammetric Engineering & Remote Sensing, vol. 72, no. 4, pp. 357–363, 2006.

F. Lafarge and C. Mallet, “Modeling urban landscapes from point clouds: a generic approach,” 2011.

S. Lang and T. Blaschke, “Hierarchical object representation-comparative multi-scale mapping of anthropogenic and natural features,” INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES, vol. 34, no. 3/W8, pp. 181–186, 2003.

J. Liu, J. Shen, R. Zhao, and S. Xu, “Extraction of individual tree crowns from airborne LiDAR data in human settlements,” Mathematical and Computer Modelling, vol. 58, no. 3, pp. 524–535, 2013.

R. A. MacMillan and P. A. Shary, “Landforms and landform elements in geomorphometry,” Developments in soil science, vol. 33, pp. 227–254, 2009.

D. Mongus and B. Žalik, “Parameter-free ground filtering of LiDAR data for automatic DTM generation,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 67, pp. 1–12, 2012.

Mongus, Domen, and Borut Zalik. “Computationally Efficient Method for the Generation of a Digital Terrain Model from Airborne LiDAR Data Using Connected Operators.” Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of 7, no. 1 (2014): 340–351.

M. Musci, R. Q. Feitosa, and G. A. Costa, “An object-based image analysis approach based on independent segmentations,” in Urban Remote Sensing Event (JURSE), 2013 Joint, 2013, pp. 275–278.

M. Riesenhuber and T. Poggio, “Hierarchical models of object recognition in cortex,” Nature neuroscience, vol. 2, no. 11, pp. 1019–1025, 1999.

A. J. Rodríguez-Sánchez, M. Fallah, and A. Leonardis, “Editorial: Hierarchical Object Representations in the Visual Cortex and Computer Vision,” Frontiers in computational neuroscience, vol. 9, 2015.

C. Sevara, M. Pregesbauer, M. Doneus, G. Verhoeven, and I. Trinks, “Pixel versus object — A comparison of strategies for the semi-automated mapping of

archaeological features using airborne laser scanning data," Journal of Archaeological Science: Reports, vol. 5, pp. 485–498, Feb. 2016.

G. Sithole and G. Vosselman, "Report: ISPRS comparison of filters," ISPRS commission III, working group, vol. 3, 2003.

G. Sithole, Segmentation and classification of airborne laser scanner data. TU Delft, Delft University of Technology, 2005.

G. Sohn, X. Huang, and V. Tao, "Using a binary space partitioning tree for reconstructing polyhedral building models from airborne lidar data," Photogrammetric Engineering & Remote Sensing, vol. 74, no. 11, pp. 1425–1438, 2008.

Smith, M.J., P. Paron, and J.S. Griffiths, Eds., 2011. "Geomorphological Mapping: Methods and Applications". Amsterdam, Boston, Heidelberg, London, New York, Oxford, Paris, San Diego, San Francisco, Singapore, Sydney, Tokyo, Elsevier Science, 612p.

S. Sun and C. Salvaggio, "Aerial 3D building detection and modeling from airborne LiDAR point clouds," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 6, no. 3, pp. 1440–1449, 2013.

H. Tang et al., "A multiscale latent dirichlet allocation model for object-oriented clustering of VHR panchromatic satellite images," IEEE Transactions on Geoscience and Remote Sensing, vol. 51, no. 3, pp. 1680–1692, 2013.

Toscano, G., Acharjee, P., McCormick, C. & Devarajan, V., (2015). Water Surface Elevation Calculation Using LiDAR Data. Proceedings, ASPRS Conference, Tampa, Florida

D. Tóvári and N. Pfeifer, “Segmentation based robust interpolation-a new approach to laser data filtering,” International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Enschede, Netherlands, vol. 36, no. 3/W19, pp. 79–84, 2005.

C. J. Van Rijsbergen. “Information Retrieval”, 2nd edition. Dept. of Computer Science, University of Glasgow, 1979.

V. Verma, R. Kumar, and S. Hsu, “3D building detection and modeling from aerial LIDAR data,” in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), 2006, vol. 2, pp. 2213–2220.

G. Vosselman, B. G. Gorte, G. Sithole, and T. Rabbani, “Recognising structure in laser scanner point clouds,” International archives of photogrammetry, remote sensing and spatial information sciences, vol. 46, no. 8, pp. 33–38, 2004.

M. Weinmann, S. Urban, S. Hinz, B. Jutzi, and C. Mallet, “Distinctive 2D and 3D features for automated large-scale scene analysis in urban areas,” Computers & Graphics, vol. 49, pp. 47–57, 2015.

M. F. Wu, Z. C. Sun, B. Yang, and S. S. Yu, “A Hierarchical Object-oriented Urban Land Cover Classification Using WorldView-2 Imagery and Airborne LiDAR data,” in *IOP Conference Series: Earth and Environmental Science*, 2016, vol. 46, p. 1, 2016.

S. Xu, G. Vosselman, and S. O. Elberink, “Multiple-entity based classification of airborne laser scanning data in urban areas,” ISPRS Journal of photogrammetry and remote sensing, vol. 88, pp. 1–15, 2014.

W. Yao, S. Hinz, and U. Stilla, “Object extraction based on 3D-segmentation of lidar data by combining mean shift with normalized cuts: Two examples from urban areas,” in 2009 Joint Urban Remote Sensing Event, 2009, pp. 1–6.

W. Yao, S. Hinz, and U. Stilla, “Automatic vehicle extraction from airborne LiDAR data of urban areas aided by geodesic morphology,” Pattern Recognition Letters, vol. 31, no. 10, pp. 1100–1108, 2010.

J. Zhang, X. Lin, and X. Ning, “SVM-based classification of segmented airborne LiDAR point clouds in urban areas,” Remote Sensing, vol. 5, no. 8, pp. 3749–3775, 2013.

J. Zhang, M. Duan, Q. Yan, and X. Lin, “Automatic vehicle extraction from airborne LiDAR data using an object-based point cloud analysis method,” Remote Sensing, vol. 6, no. 9, pp. 8405–8423, 2014.

X. Zhu and T. Toutin, “Land cover classification using airborne LiDAR products in Beauport, Québec, Canada,” International Journal of Image and Data Fusion, vol. 4, no. 3, pp. 252–271, 2013.

Appendix A

The Mathematical formulae used to compute IDW for raster interpolation

The IDW for the raster interpolation is expressed below with referring to Figure 88:

$$Z_r = \frac{\sum_{n=1}^{N} \frac{Z_n}{d_n}}{\sum_{n=1}^{N} \frac{1}{d_n}} \qquad \textit{eq. 16}$$

Where, Z_r = interpolated elevation value

Z_n = Surface elevation at sample point

d_n = distance from sample point to center

N = number of sample points

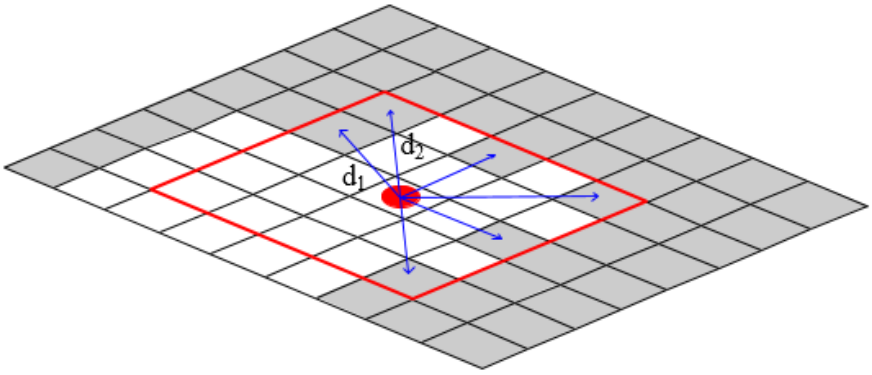


Figure 88 IDW for raster interpolation

Appendix B

Plane fitting algorithm adapted from (Tóvári and Pfeifer, 2005)

The planar segmentation is based on a region growing algorithm, adapted from (Tóvári and Pfeifer, 2005), where the seeds are placed first randomly and then in unexplored spaces to divide the target area into planar segments. Once a seed triangle is selected, the nearest neighbor cells are examined to determine whether they meet certain criteria. An optimal plane is estimated from the sample points representing the cells in the growing region. There are two popular approaches to find optimal planes: Least square fitting and principal component analysis. In our method, we used the principal component analysis since it has been proven to be more robust in the presence of noise (Huang and Tseng, 2008). The PCA computes the eigenvalues and the eigenvectors of the sample covariance matrix. The largest two eigenvectors give the dimension of the plane. Hence the cross-product of the two vectors give the normal of the plane.

Nearest neighbor triangles are incorporated into the growing region if they fulfill the following three criteria:

1. the angle difference between the normal vectors of the optimal plane and the neighbor triangle is below a certain threshold value,
2. the distance from the optimal plane to the triangle is shorter than a predefined maximum value,
3. the distance between the center of the optimal plane and the neighbor triangle is under a threshold value.

The optimal plane is re-estimated after each time a triangle is accepted to the growing region. The growing continues until there is no triangle left.

Appendix C

Robust interpolation algorithm adapted from (Tóvári and Pfeifer, 2005)

The plane segments extracted from the plane fitting algorithm (described in Appendix B) is feed into the robust interpolation algorithm that enables filtration of the segments belong to the ground. First, planar segments with a size larger than the largest man-made structure possible are extracted and identified as ground. These ground seeds act as ground elevation references to initiate the filtering process.

During the iterative process, each point (x_j, y_j, z_j) within the segment has one weight w_j. This weight is initially set to one. The robust interpolation algorithm runs as follows:

1. A surface is interpolated initially, using the surface moving least squares (MLS) with an order one polynomial (a plane) and the current weights (w_j) of the points associated with the segments. The current weight of the points is further adjusted via its product to a 2-dimensional weight function which ensures that points near the interpolation position gain higher weights and reach a value of zero at a certain range.

2. For each point, filter value (r_j) is determined by the difference in the distance of the interpolated surface to the constituents observed value.

3. Since a segment can either entirely belong to ground, or entirely belong to an object, all filter value belonging to each segment are analyzed together. A representative filter value (R_k) is therefore determined by averaging the filter values

belonging to each segment. A new weight (W_k) is assigned based on this aggregated value and a weight function for robust adjustment. The function to assign a weight (Wk) for the segment used in (Tóvári and Pfeifer, 2005) is shown in Figure 89

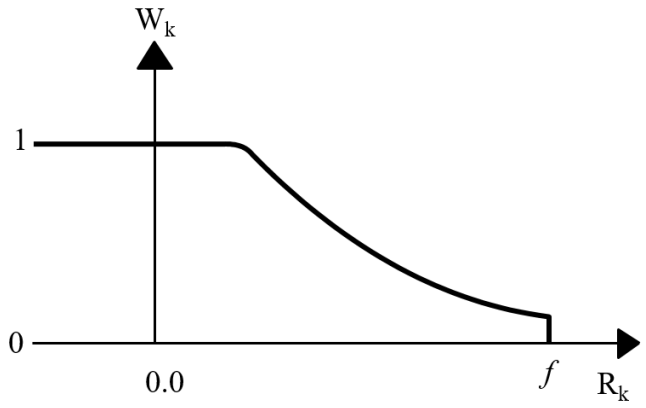


Figure 89 Parameters of the weight function

Here, the weight drops from the maximum 1 to 0 as R_k value increases from 0 to f. However, the weight remains constant at 1 for all negative filter values, i.e. segments with an average filter value less than zero will always have the maximum weight. This means the interpolation is forced to conform to surface that lies below the interpolated surface. Therefore, planar segments of objects that are convex in shape are filtered out and those that belong to concave objects are ignored. However, as stated in chapter 1, my objective is to detect not only convex objects but also objects that are concave in shape such as a pond, artificial cavities. To serve our purpose, the weight function is modified as shown in Figure 90. Here, the left branch is a mirror image of the right branch. Therefore, this

enables the interpolation algorithm to filter out planar segments that belong to both concave and convex.

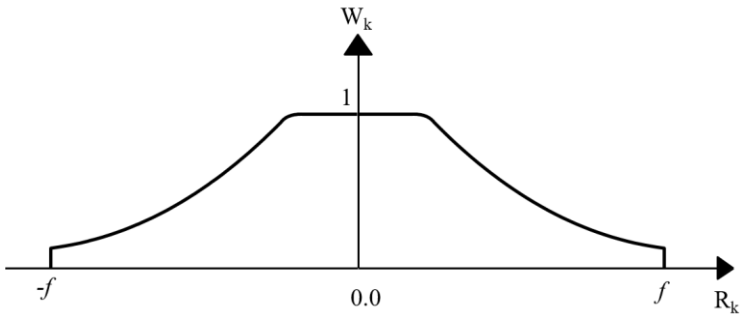


Figure 90 Parameters of the modified weight function

4. These weights are considered in the next iteration and therefore, segments with a large weight have a larger influence on the run of the surface. This process is iterated until there is no object point left, i.e. their weight becomes zero.

Biography

Md. Ahsan Habib received his B.Sc. degree in Electrical and Electronic Engineering (EEE), in 2009 from the University of Dhaka, Dhaka, Bangladesh. He earned his Ph.D. in Electrical Engineering at the University of Texas at Arlington, Arlington, Texas in 2017. He has been awarded a Doctoral Dissertation Fellowship in Summer 2017 from UT Arlington. His research interests are in Computer Vision, Machine Learning, and Robotics.