A Cluster-based Morphological Filter for Geospatial Data Analysis

[1]Zheng Cui, [2]Keqi Zhang, [3]Chengcui Zhang, [1]Shu-Ching Chen
[1]School of Computing and Information Sciences, Florida International University, USA
[2]Department of Earth and Environment, Florida International University, USA
[3]Department of Computer and Information Sciences, The University of Alabama at Birmingham, USA

[1,2]{cuiz, zhangk, chens}@fiu.edu
[3]zhang@cis.uab.edu

ABSTRACT
LIDAR (Light Detection and Ranging) is a widely used technology to measure terrain properties and topographic mapping nowadays. Many filtering methods have been developed to process the geospatial data generated by LIDAR to generate bare earth digital terrain models. Among these methods, mathematical morphological filtering is a very effective and efficient method to separate ground and non-ground objects from LIDAR data. This method can achieve ideal results in the flat terrain, while it is not working very well in the undulating and complex terrain with large non-ground objects. The reason is that it would remove ground terrain objects along with filtering large size non-ground objects when using a large filtering window size. Especially in the mountainous terrain, it would cause the hill cut-off problem, which is a common problem for morphological filters. In this paper, a cluster-based morphological filter is proposed to improve the progressive morphological filter and make it work better on more undulating and complex terrain types. The filtering results demonstrate that the proposed method is able to effectively preserve terrain ground objects and remove large non-ground objects.

Categories and Subject Descriptors
I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms; Surface and solid representations; I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism—Visible line/surface algorithms

General Terms
Algorithms.

Keywords
LIDAR, Digital terrain model (DTM), data filtering, geospatial data analytics

1. INTRODUCTION
As LIDAR (Light Detection and Ranging) has become a widely used technology in surveying and industrial measurement applications in recent years, many filtering methods have been developed. The mathematical morphological method is a widely used technique in many filters [1][2][3][4]. Kilian et al. [1] first proposed to use mathematical morphology to filter non-ground points. The disadvantage of this method is that ground points would be filtered out when the filtering window is too large. Petzold et al. [2] proposed a method which uses the moving filter window from large scale to small scale to filter the points iteratively. This method first created a rough terrain model by finding the lowest points in a relatively large size moving window. The points with elevation difference greater than the threshold were filtered out, and a more precise terrain model would be achieved. The process was repeatedly carried out along with reducing the window size until a final terrain model was achieved. The result of this method is affected by the final window size and the final threshold below which points are expected to be terrain points. Zhang et al. [4] use moving filter window from small scale to large scale to do the *open* operation [4] of mathematical morphology. The elevation difference after the *open* operation would be compared with the predefined threshold for each filter window size. The point would be filtered out if the elevation difference is over the threshold. Since many morphological filters need to use the filtering window in different sizes, a large filtering window size would cause the common cut-off problem, when filtering large and variable sizes of non-ground objects. Because when the moving window size is too large, it would remove many ground surface points as well. A small window size would not remove points on the large buildings and keep them as ground points. A relatively large window size would remove the small local discontinuities and smooth the terrain in a large scale. The elevation difference threshold would affect the result significantly. A high threshold in the final step would classify many vegetation points as ground points, while a low threshold would filter out more small terrain discontinuities. The selection of the parameters depends on the terrain types remarkably. Therefore, in this paper, a cluster analysis method is proposed to detect the terrain shape and connectivity of neighboring points so as to prevent the ground points being removed after the filtering window size is increased to a certain level.

The effectiveness of the proposed method has been evaluated on both relatively flat terrains and more undulating and complex terrains, and in both cases the proposed method has shown promising results, while several existing segmentation and cluster-based filters [7][8][9] have difficulty processing more complex terrain types [5].

BIGSPATIAL '13:, November 05 - 08 2013, Orlando, FL, USA.

http://dx.doi.org/10.1145/2534921.2534922

2. A CLUSTER-BASED MORPHOLOGICAL FILTER

A new mechanism of justifying ground and non-ground points is proposed to improve the morphological filter in this paper. This mechanism called cluster analysis method is introduced in the filtering procedure. Combining with the progressive morphological filter [4], this method will help the filtering procedure justify the ground and non-ground points identified from the progressive morphological filter and make the filtering result more accurate on a greater variety of terrain types. The progressive morphological filter would generate more errors in some undulating terrain such as mountain areas, especially if there are some relatively large non-ground objects such as large buildings or constructions in the terrain. The cause of this problem is that the progressive morphological filter would remove more ground points along with filtering large non-ground objects. The key to reduce the errors is to distinguish what kind of points removed during the filtering process are from large non-ground objects and what kind of points are from the ground surface.

The essential of the cluster analysis method is to build up some collections of ground object point clusters for each row or column of the grid data before each filtering step. These collections of clusters will be used to justify the filtering result of the corresponding row or column after each filtering step. The clusters of each row or column are actually the candidate ground surface before each filtering step, some of which could be identified as non-ground objects in the following filtering steps, but they are not known until the final filtering step (as detailed in Sections 2.2-2.3). If they are indeed ground objects, they would be preserved at the end of the filtering. Otherwise, they would be filtered out as non-ground objects eventually. After each filtering step, some non-ground objects could be identified and labeled with the filtering window size of the current filtering step. These identified non-ground objects will be justified by the clusters generated at the beginning of each filtering loop. If certain criteria are met, these identified non-ground objects will be labeled back to ground objects. That is how the cluster analysis method collaborates with the filtering procedure.

2.1 Cluster Generation

Since the proposed cluster analysis method and the morphological filter are both based on grid data structure, the input data set need to be gridded before processing. Each grid would choose a representative point from the points within the grid, and the filter treats each grid as one point. The lowest point in each grid can be used as the representative point of that grid, and the nearest neighboring grid's representative point can be used for the empty grids. The filtering status of all the grids can be represented by a mark matrix. Each unfiltered point grid's status will be initialized with unfiltered point's mark ("0" or "-7777") and stored in the *mark* matrix. "0" means real unfiltered points from the data set, while "-7777" refers to the interpolated unfiltered points.

To generate clusters for each row or column of the data grid, the data points will be scanned from the first to the last in each row or column, and unfiltered points with "0" or "-7777" marks will be collected into different clusters based on the elevation difference between consecutive unfiltered points. The way for collecting unfiltered points into the same cluster will be demonstrated on the row direction as follows. It would be working for the column in the same way. From the first unfiltered point of a row, the next unfiltered point would be continuously searched and collected into clusters. There could be two scenarios for the next unfiltered point in terms of the spatial relationship. One is that the next unfiltered point is adjacent to the previous unfiltered point; the other is that the next unfiltered point is separated by some filtered points identified in the previous filtering steps. If the two unfiltered points are next to each other, the elevation difference between these two points will be compared with the predefined cluster threshold to determine whether they should be separated into different clusters. If the elevation difference of these two points is less than a predefined threshold for separating clusters, they will be collected into the same cluster, otherwise a new cluster will be created and the following unfiltered point will be collected into the new cluster. While in the second scenario, the next unfiltered point is separated by some filtered points, and the criteria of separating clusters would be different. The slope of the two unfiltered points' elevation will be compared with a predefined threshold. To simplify these two cases, the same cluster threshold value is used. Thus, the cluster threshold would be a slope threshold for both scenarios, which is shown in Line 8 of the algorithm description. Different thresholds could be given for the two scenarios if needed. By this means, it will generate a collection of clusters that represents the unfiltered points in each row or column of the grid.

Figure 1 shows the result of cluster generation in a row. Points within the same cluster are connected by red lines. There is no line connecting different clusters.

For each row or column of data, there could be some unfiltered points during each filtering step. The cluster analysis method will generate a collection of clusters based on those unfiltered points in each row or column. These clusters are treated as candidate ground point clusters, because some of them are on the ground surfaces, while others are on the non-ground objects surfaces. The algorithm of cluster generation is described as follows.

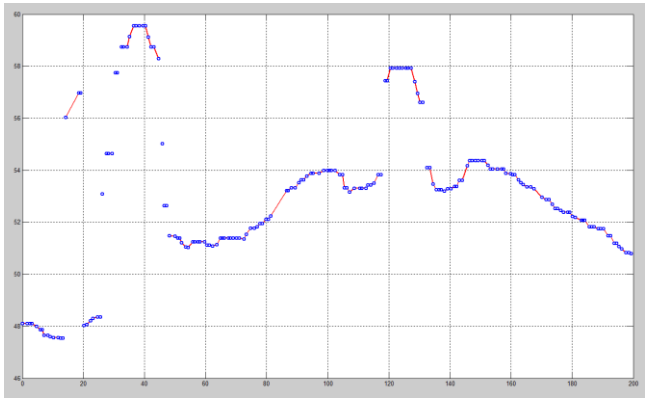


Figure 1. Clusters of data points

Algorithm description: *MakeCluster*

INPUT:
1. An array of z value for a row or column: *rz*
2. Threshold to separate different clusters: *clusterThreshold*

OUTPUT:
An array of cluster struct: *clusters*

1. Scan array to find the first unfiltered or interpolated unfiltered point and save the index in prevGrd
2. If prevGrd is NULL then return empty *clusters* array
3. j←1 // Initial cluster index
4. *clusters*(1) = *CreateTrendCluster*(prevGrd)
5. for i=(prevGrd+1)→ n // n is the size of a row or column
6. if *rz*(i) is a ground or interpolated ground point then

```
7.          preDiff = rz(i)-rz(prevGrd)
8.          if(abs(preDiff/(i-prevGrd))>clusterThreshold) then
9.              clusters(j).last = i-1; // New cluster is found, record
                the ending index for the previous cluster
10.             clusters(j).postDiff = -preDiff; // Store elevation
                difference between the last point of the current
                cluster and the first point of the next cluster
11.             j = j+1;          // Increase the cluster index
12.             clusters(j) = CreateTrendCluster(i); // Create a new
                cluster starting from rz(i)
13.         end if
14.       end if
15.       prevGrd ← i
16.   end for
```

The *MakeCluster* algorithm shows that all the points in a row or column will be scanned one after another, and checked whether they can be collected into the same cluster. The prevGrd is used to store the index of the latest found unfiltered point. If no unfiltered or interpolated unfiltered points were found, it would return an empty cluster array in Line 2. Otherwise, at least one cluster would be generated. *CreateTrendCluster* procedure is for generating and initializing a cluster struct. The first unfiltered point's index will be stored when the cluster was created in Line 4. The algorithm will continually scan the input row or column data for the next unfiltered point. If the slope formed by the elevation difference between the two unfiltered points is greater than the cluster threshold, a new cluster will be identified and created. All the unfiltered points that meet the cluster criteria will be collected into the same cluster until a new cluster is found. The index of the last unfiltered point in the same cluster will be recorded in Line 9. The elevation difference between the last point of the current cluster and the first point of the next cluster will be recorded in the current cluster's struct.

After scanning the whole row or column data, one or multiple clusters would be created if the row or column contains unfiltered points. Each cluster stores the indexes of all the unfiltered points that belong to it and some other information, such as elevation difference between neighboring clusters. This information will be used in the cluster analysis procedure.

2.2 A Cluster-based Morphological Filter

In this paper, a cluster-based morphological filter is proposed by combining cluster analysis method with the progressive morphological filter [4]. This new filter can effectively prevent the terrain surface points from being removed when the filtering window size is increased to a certain level.

The structure of this cluster-based morphological filter is based on the progressive morphological filter [4]. The input data has to be gridded with a certain grid size and interpolated for the empty grids with some interpolation method such as the nearest neighbor interpolation. During each step of progressive morphological filtering, a moving window will be used in the *open* operation which includes *erosion* and *dilation* operations. After the *open* operation, the elevation difference of each corresponding pair of grids will be calculated between the two surfaces which are formed before and after the *open* operation. The elevation difference of each grid pair will be compared with the predefined threshold of the current filtering step to determine whether the point is a non-ground point. If the elevation difference is greater than the threshold, the point would be classified as a non-ground point and marked with the current filtering window size value. In the proposed method, after the filtering window size is increased to a certain level, a collection of clusters for each row or column will be generated before *open* operation. The cluster analysis will be carried out to justify the *mark* matrix result at the end of the filtering loop. The unfiltered point clusters are used to represent the candidate terrain surface based on the unfiltered points prior to the current filtering step. A mark matrix is used to record the filtering status for data grids. The algorithm description of this cluster-based morphological filter is as follows.

Algorithm description: *MorphClusterFilter*

INPUT:
1. Gridded and interpolated LIDAR data: *Z*
2. Progressive morphological filtering Steps: *steps*
3. Initial cluster windows size to start cluster analysis: *cluster_window_size*
4. Threshold to separate different clusters: *clusterThreshold*
5. Morphological filtering threshold array: *threshold*
6. Morphological filtering window size array: *window_size*

OUTPUT:
Mark Matrix which represents the filtering result: *mark*

```
1.  Initialize mark matrix with "0"  or "-7777" marks
2.  for k = 1 to steps
3.    for each direction (by row and by column)
4.      for each row/column in grid
5.        zcurr ← Zi // extract the ith row/column's z values
6.        if window_size(k)>= cluster_window_size then
7.          clusters ← MakeCluster(zcurr, clusterThreshold)
8.        end if
9.        zopen ← Morphopen(zcurr, window_size(k));// Open
                                                    operation
10.       zdiff ← zcurr – zopen;
11.       if zdiff(i, j) > threshold(k) then
12.         mark(i, j) ← window_size(k); // Update mark
13.       end if
14.       if window_size(k)>= cluster_window_size then
15.         UpdateMark(mark, clusters);  // Update mark
16.       end if
17.       Zi ← zopen; // Set the ith row/column's z values
18.     end of each row/column
19.   end of for loop of each direction
20. end of for loop of all steps
```

At the beginning of the algorithm in line 1, the *mark* matrix is initialized with all "0" or "-7777" marks, which means that all the points are treated as unfiltered points at first. "0" means real points in the input data, while "-7777" represents interpolated points. Both marks represent unfiltered data points. The points' marks could be changed in multiple steps of filtering. The steps of the filter, the filtering window size of each step, and the threshold associated with each window size are predefined by the user. Users can also control when the cluster analysis will be activated in the filter by defining *cluster_window_size* (more details on this to follow in Section 2.3). In Line 6 of the algorithm, after the filtering window size is greater than the predefined *cluster_window_size*, the cluster analysis will be activated in the filter. *MakeCluster* procedure will generate the clusters of unfiltered points for each row or column. *Morphopen* will carry out the *open* operation which includes *erosion* and *dilation* operations. The elevation difference between the two surfaces that are acquired before and after *open* operation will be compared with a predefined threshold and used to update the *mark* matrix accordingly for the filtered points. The *UpdateMark* procedure

will perform cluster analysis to find whether there is any point that is filtered out in the current step that can be reset to unfiltered points.

In the filtering procedure, the *mark* matrix represents each grid's filtering status, if the point in the grid is identified as a ground point, it will be set as the ground point's mark value ("0"); if the point is identified as a non-ground point, it will be marked with current filtering step's windows size which is a positive value. Therefore, a point can be identified as a non-ground point by the mark that reflects the filtering window size. It also shows in which filtering step the point was filtered out. An interpolation mark with a negative value is used to represent the interpolated points for the empty grids. If the interpolated point is identified as non-ground point, it will be marked as the negative value of the current filtering window size which is always greater than -7777. The real data points and interpolated points can be easily distinguished in this way.

The initial mark matrix contains only two kinds of values which are either "0" or interpolation mark value ("-7777"). If the initial mark is not changed until the end of the filtering process, the point with "0" or interpolation mark value indicates the point was classified as ground point. In each morphological filtering loop, the mark matrix may contain up to four kinds of values, as summarized in Table 1.

Table 1. Mark values and their meanings

Mark value	Meaning
0	Before filtering: Real data point After filtering: Real ground point
-7777	Before filtering: Interpolated data point After filtering: interpolated ground point
Positive window size	Real filtered point (non-ground point)
Negative window size	Interpolated filtered point (interpolated non-ground point)

Since the proposed method is based on progressive morphological filtering, an increasing window size is used in each filtering step [4]. Therefore, points filtered in each step will be marked with the window size. Real filtered points and interpolated filtered points will be marked with the positive and negative window size values, respectively. This labeling strategy makes it easy to track in which step the points were filtered out, which reveals at what scale the points were removed (as non-ground points). It provides more information for the cluster analysis. Furthermore, this labeling strategy is also very helpful for outputting the filtering result, because real ground points and interpolated ground points can be easily separated in this way.

Before each morphological filtering step, there could be some unfiltered points in each row or column. The proposed method will generate a collection of unfiltered point clusters based on each row or column's mark values and their elevation differences. These clusters will help justify the marks of points filtered out in this filtering step.

2.3 Cluster Analysis

The main idea of cluster analysis method is to use the cluster information to check whether the filtered points in each filtering step are on the candidate ground surface. If these filtered points that have the same filtering window marks are completely falling into an unfiltered point cluster, the filtered points' marks will be reset to unfiltered mark ("0" or "-7777"). However, since these reset points are still candidate ground points, they could still be filtered in later steps.

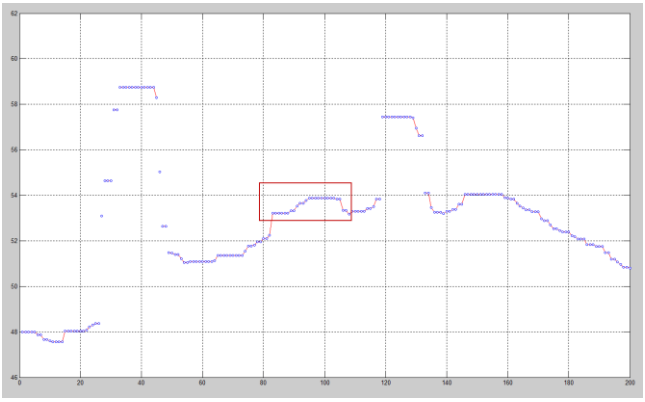


Figure 2. Filtered points in a cluster

A sample cluster collection is shown in Figure 2. These clusters represent all the candidate ground points (in one particular row) which include points that could be later identified as non-ground objects. The points of each cluster are connected by red lines. There is no line connecting different clusters, which makes it easy to distinguish different clusters from the display. As shown in Figure 2, if the points in the red rectangle area are later classified as non-ground object points in the subsequent filtering step, their marks will be set as the corresponding filtering window size, either positive or negative, depending on the data point type (real data point or interpolated). Then the *UpdateMark* procedure in Line 15 (Algorithm: *MorphClusterFilter*) will check whether the range of these consecutive filtered points marked with the current filtering window size are completely falling into the range of a unfiltered point cluster identified immediately before the filtering step. As in the figure, they are completely located in a single cluster. Therefore, their marks will be reset to ground point's mark before the next clustering-filtering iteration. In this way, potential ground points can be preserved on the ground surface successfully and avoided to be removed when the filtering window size is too large. Since the cluster analysis is embedded in the progressive morphological filter's loop, the points whose marks have been reset to "0" or "-7777" could still have a chance to be filtered out in later filtering steps. This makes it possible for those real non-ground objects to be removed in some later steps, even though they might be treated as ground points in the previous steps. In fact, all the points with ground point's mark ("0" or "-7777") are actually candidate ground points until the end of the filtering, by then their final identities will be decided after the last filtering step. The cluster analysis algorithm is listed as follows. It uses the binary search to check whether the filtered points' range is completely falling into the range of a single unfiltered point cluster.

Algorithm description: *UpdateMark* for cluster analysis

INPUT:
1. Mark matrix: *mark*
2. Clusters generated before *open* operation: *clusters*

OUTPUT:
Updated mark matrix: *mark*

1. Scan marks in the current row or column and find the ranges of consecutive filtered point segments with the mark value of the current filter window size.
2. Check each range of filtered point segment with that of the clusters by using binary search.
3. if the range of a filtered point segment is completely falling into a single clusters then
4. Reset the marks in the range to unfiltered marks
5. end if

The cluster analysis can be involved in any step of the filter. Since the morphological filter will not result in significant errors under small filtering window sizes, the cluster analysis method can be activated after the filtering window is increased to a certain size. After that, a collection of clusters for each row and column will be generated before the *open* operation in each filtering step. After *open* operation, the *mark* matrix will be updated based on the morphological filtering result, which reflects the filtering status in this step. The cluster analysis procedure will then scan each row or column's marks in order to find one or more sets of consecutive non-ground points marked with the value of the current filtering window size, meaning that they are filtered out during the latest filtering step. During the scan, only the points marked with the latest step's filtering window size value will be checked, which could prevent the classified non-ground points in prior steps from being reset in the *UpdateMark* procedure. The filter can easily distinguish the filtered points by their marks. Further, the ranges of consecutive filtered point segments will be compared with that of the clusters generated before the *open* operation in each filtering step to check whether they are completely falling into any unfiltered point cluster. Each collection of consecutive filtered points, i.e., a segment, is represented by the starting and the ending indexes of the consecutive points, which represent a range in the *mark* matrix. The unfiltered point cluster created before the *open* operation is also represented in the same way. If a consecutive filtered point segment falls into any unfiltered point cluster identified in the latest clustering step, their marks will be reset with the ground point's mark value ("0" or "-7777"), which means that they will be treated as ground points in the following filtering steps or in the final result. In this way, it can prevent those large ground objects from being removed by progressive morphological filtering, while allowing progressive morphological filtering to filter out large size non-ground objects.

2.4 Cluster Threshold (*clusterThreshold*)

The main parameter for the cluster analysis method is the predefined cluster threshold. This threshold is used to separate discontinuous clusters. Since this threshold is used to distinguish potential continuous surface, it does not need to be very accurate as long as it can separate terrain surfaces into different layers. The threshold value relies on the data resolution, terrain types, and non-ground features' shapes. The discontinuity of ground surface would be different on varied terrain types, making the selection of cluster thresholds slightly different. However, the elevation differences between ground surface points and non-ground surface points, especially in the case of large size non-ground objects, are almost always significant. Thus, an approximate cluster threshold would be able to separate ground surface points and non-ground surface points into different layers by grouping points into different clusters, which makes the use of a relatively fixed cluster threshold value suitable for various terrain types. Normally, a cluster threshold between 0.5 and 1 meter on a 1 meter grid size resolution data set can work well on many different terrain types. This cluster threshold is actually a slope threshold as shown in the algorithm description of *MakeCluster* (Line 8). Users can define their own cluster threshold based on the data set's grid resolution, terrain variations, and non-ground objects' shapes.

2.5 Computational Complexity

Based on the algorithm description of the proposed cluster-based morphological filter, the time complexity of the algorithm should be the same as that of the progressive morphological filter. Since the proposed filter is based on the progressive morphological filter, the major computation time for the progressive morphological filter is the *erosion* and *dilation* operation in addition to the interpolation. The time complexity for the *open* operation is O(wN), where w is the window size of the morphological filter and N is the number of grids which is the product of the number of rows and columns. For M windows, the time complexity is equal to [4].

$$O(\sum_{k=1}^{M} w_k N)$$

Since in the cluster generation procedure *MakeCluster*, the whole data set just need to be scanned once to generate all the clusters, the time complexity of cluster generation is O(N). In the cluster analysis procedure, the time complexity of cluster search for each reset range is O(log^N) based on a binary search, therefore, the overall time complexity of the cluster-based morphological filter is the same as the progressive filter. The space complexity of the proposed algorithm is O(N), because the space complexities of the *mark* matrix and the *clusters* are both O(N).

3. EXPERIMENT RESULTS ANALYSIS

Filtering experiments have been carried out in the sample data set of a mountain area in California. One of the testing data set is an undulating terrain with various sizes of non-ground objects such as trees, vegetation, and buildings. The data set covers a 200 x 200 square meters area. The grid size is 1 meter, which gives a 200 x 200 grid mesh. The original data set's 3D mesh diagram is shown in Figure 3.

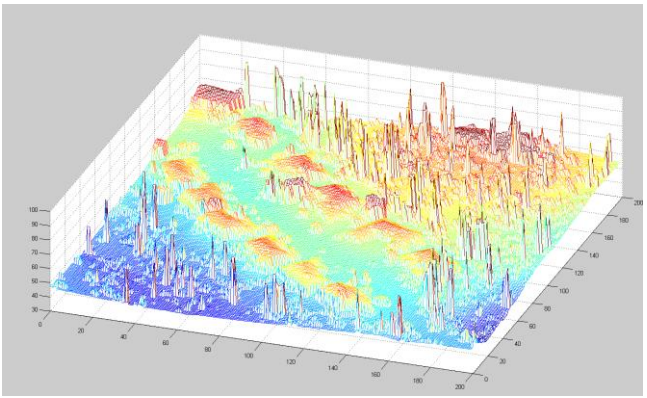
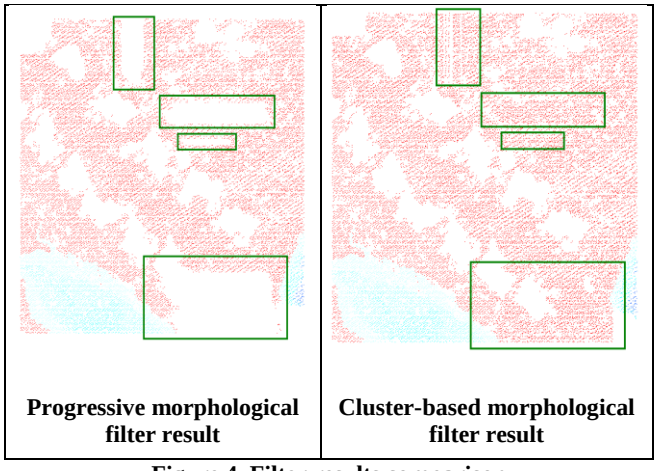


Figure 3. The original data set 01 3D mesh

First, the progressive morphological filter was used to filter the data set. A half window size is used as each step's filtering window size parameter. The half window size is the number of grids extending to the left or right from the current grid. In our case, the half window size series for each filtering step is 1, 2, 4, 8, 16, 20, and 32, which makes the total number of filtering steps 7. It is worth noting that the number of steps depends on the sizes of non-ground objects in the data set. Normally using window sizes 1, 2, 4 are sufficiently effective for removing small and intermediate objects. Whether or not to go for larger window sizes will be determined by the sizes of non-ground features. The results of applying the two filters are shown in Figure 4. The filtering results show that the progressive morphological filter mistakenly removed a substantial amount of the ground surface points in several areas. Those areas are shown in the green rectangles. The progressive morphological filter's result demonstrates a common problem of many existing morphological filters. When the filtering window size is increased to a certain level, the ground points will inevitably be removed by the filter. That is the intrinsic problem from the morphological filter itself. This problem would be aggravated on undulating and complex terrains, such as mountain areas. As in this testing data set, there are several buildings located on the mountain slope, making the cut-off problem more apparent. However, if only smaller filtering window sizes were used in the filter, it would have incorrectly identified some buildings as ground points. This would be a trade-off between the omission and commission errors of the filter. To alleviate this kind of problem, some other mechanisms which can detect the connectivity and smoothness of the ground surface have to be incorporated. The proposed cluster analysis method will assist to lower the omission and commission errors at the same time. As shown in the filtering results, the cluster-based morphological filter can preserve the ground surface effectively, especially when the filtering window size is increased to a certain level. Compared with the result of progressive morphological filter, the proposed method not only preserved the ground points on the mountain slope, but also filtered out the buildings successfully.



Progressive morphological filter result | **Cluster-based morphological filter result**

Figure 4. Filter results comparison

The mountain area is a typical terrain type which would cause the cut-off problem for morphological filters. However, even in relatively flat terrains, this kind of cut-off problem could happen if the filtering window size needs to be increased to a relatively large level in order to remove large non-ground objects, such as large buildings and huge constructions. Another test data set is from a relatively flat terrain with complex buildings, which can demonstrate this situation. The original data is shown in Figure 5. There are several complex shape buildings which are displayed in the red points. There are some relatively low points on the ground. In this situation, when the filter window size reaches a certain level, it could cut off a substantial amount of ground points.

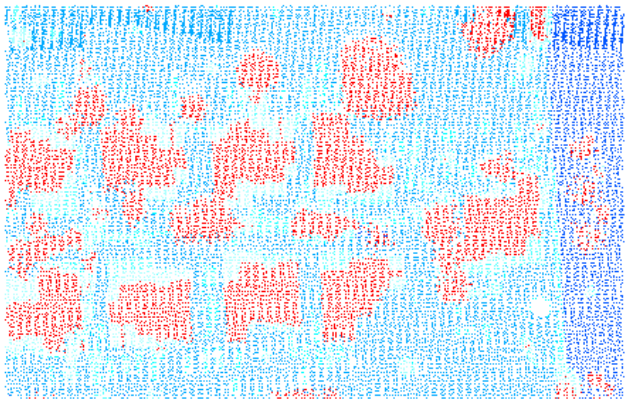


Figure 5. The original data set 02

Figure 6. Progressive morphological filtering result with the cut-off problem

The filtering result from the progressive morphological filter is shown in Figure 6. The empty white areas are the cut-off areas that include not only the non-ground objects but also some ground points. The result shows that the large filtering window size would make the morphological filter remove some ground surface along with large buildings, even for relatively flat terrains. This case demonstrates a trade-off situation for morphological filter. If the large buildings need to be completely removed, the filtering window size has to be large enough, potentially causing the cut-off problem. However, if smaller window sizes were used, it may not be able to successfully remove large non-ground objects. Therefore, the proposed filter would help in this situation in that it would allow the morphological filter to use large filtering window size while preserving the ground surface points. This benefits from the recovery mechanism from the cluster analysis method, because it can reset some of the removed points' labels from filtered marks to unfiltered marks by checking their spatial relationship with the previously identified unfiltered point

clusters, if certain criteria are met. Figure 7 is the filtering result of the proposed filter. The result shows that the buildings were successfully removed, while the ground points were successfully preserved.

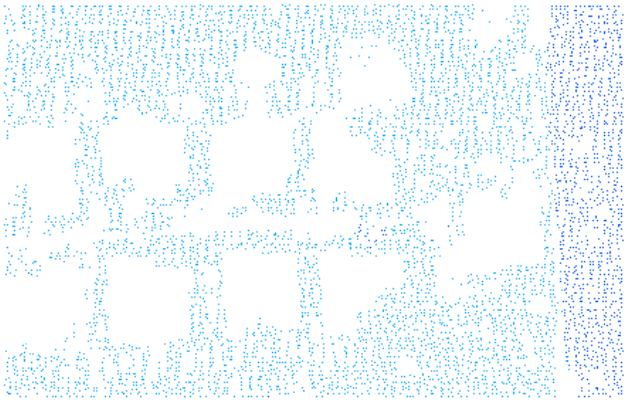


Figure 7 Cluster-based morphological filtering result

4. CONCLUSIONS AND FUTURE WORK

Experiments results show that the proposed cluster-based morphological filter is able to improve the progressive morphological filter on undulating and complex terrains significantly. Especially when the filtering window size is increased to a relatively large size, it would prevent the ground terrain from being removed while still able to filter out large size non-ground objects.

The cluster analysis can be extended to two dimensional analysis with further 2D refinement of the cluster recovery mechanism.

The selections of the filtering window size and threshold parameters are very critical and sensitive to different terrain types. It would be very helpful to analyze the study area's terrain types and non-ground features for choosing the appropriate parameters. Therefore, this method still needs to involve some human effort. Some adaptive parameter selection methods might be developed to automatically choose the appropriate parameters for different terrain types, which would make this method more efficient and robust.

5. ACKOWNLEDGMENTS

We would like to thank NCALM and NOAA Florida Hurricane Alliance Research Program for providing the necessary financial support for this research.

6. REFERENCES

[1] Kilian, J., Haala, N., and Englich, M. 1996. Capture and Evaluation of Airborne Laser Scanner Data. *International Archives of Photogrammetry and Remote Sensing* (1996), Vol. XXXI, Part B3, Vienna, 383–388.

[2] Petzold, B., Reiss, P., and Stossel, W. 1999. Laser scanning-surveying and mapping agencies are using a new technique for the derivation of digital terrain models. *ISPRS J. Photogrammetry and Remote Sensing* (1999), 54, 95-104.

[3] Morgan, M. and Tempfli, K. 2000. Automatic Building Extraction from Airborne Laser Scanning Data. *International Archives of Photogrammetry and Remote Sensing.* Amsterdam (2000), 33(B3): 616-623.

[4] Zhang, K., Chen, S.-C., Whitman, D., Shyu, M.-L., Yan, J., and Zhang, C. 2003. A progressive morphological filter for removing nonground measurements from airborne LiDAR data. *IEEE Transactions on Geoscience and Remote Sensing* (April, 2003), 41, 4, 872-882.

[5] Meng, X., Currit, N., and Zhao, K. 2010. Ground filtering algorithms for airborne LiDAR data: a review of critical issues. *Remote Sensing* (March, 2010), 2, 3, 833-860.

[6] Liu, X. 2008. Airborne LIDAR for DEM generation: some critical issues. *Progress in Physical Geography* (February, 2008), 32, 1, 31-49.

[7] Filin, S. 2002. Surface clustering from airborne laser scanning data. *International Archives of Photogrammetry and Remote Sensing. Spatial Inf. Sci*. (2002), XXXIV, Part 3, 119-124.

[8] Jacobsen, K. and Lohmann, P. 2003. Segmented filtering of laser scanner DSMs. In *Proceedings of the ISPRS Working Group III/3 workshop "3-D Reconstruction From Airborne Laserscanner and InSAR Data"* (Dresden, Germany, October, 2003).

[9] Tóvári, D. and Pfeifer, N. 2005. Segmentation based robust interpolation–a new approach to laser filtering. In *Proceedings of the ISPRS Workshop Laser Scanning* (Enschede, Netherland, September 12-14, 2005), *International Archives of Photogrammetry and Remote Sensing. Spatial Inf. Sci*. 36, 79-84.