Hinged, pseudo-grid triangulation method for long, near-linear cliff analyses

Michael J. Olsen[1], *Falko Kuester* [2], *and Elizabeth Johnstone*[3]

1. PhD., M.ASCE. Assistant Professor, School of Civil and Construction Engineering, Oregon State University, 220 Owen Hall, Corvallis, OR, 97331, USA. michael.olsen@oregonstate.edu
2. PhD., Associate Professor, Department of Structural Engineering, University of California San Diego, 9500 Gilman Dr, MC 0085, La Jolla, CA, 92093, USA. fkuester@ucsd.edu
3. PhD Candidate, Scripps Institution of Oceanography, University of California San Diego, 9500 Gilman Dr, MC 0244, La Jolla, CA, 92093, USA. eajohnst@ucsd.edu

Abstract

LIDAR scanners can rapidly collect high resolution, cm-level accurate point clouds representing topography, suitable for change detection if scans are repeated over time. To perform meaningful volumetric change analyses, point clouds are commonly triangulated to produce continuous, digital terrain models (DTMs). However, DTM creation methods generally require a fixed look direction tied to a specific plane, which results in less than ideal triangulations when modeling areas with largely varying topography, such as coastal cliffs and beaches. Furthermore, for accurate volumetric change analysis, surfaces must be free of intersecting triangles, have consistent facet normal orientations, and be free of data gaps (holes). The methodology presented herein produces continuous surfaces without inconsistent normals and minimizes holes and self intersections. The few intersecting triangles and holes may be quickly repaired using existing algorithms and were shown to be substantially less abundant compared to common surfacing techniques. Finally, the data structuring of this technique significantly shortens processing time, reduces memory requirements, and enables efficient and interactive visualization through both sub-sampling at varying scales and optimized view frustum calculations.

Keywords: LIDAR, laser scanning, triangulation, surface modeling, 3D, coast, cliff, terrain

Introduction

Three-dimensional (3D) surface modeling through triangulation provides a mechanism for interpolation and analysis of 3D point cloud data. Because of recent advances in Light Detection and Ranging (LIDAR) technology, point cloud datasets tend to be quite large and include multiple adjacent scans from different viewpoints; thus, optimizations to the triangulation process can enable more efficient and accurate data analyses. Some applications, such as cliff erosion studies, evaluate change between datasets, which requires both spatial and temporal models for analysis. In Southern California, several repeat LIDAR surveys have been completed for 17 km of seacliffs along a coastline (Olsen *et al.* 2009 and 2011), which are used for sediment budget analysis. For accurate volume calculations, surface models need to be free of intersections, inconsistent facet normal orientations, and holes. Because of scanner noise and complexity of the topography in coastal environments, the creation of clean, detailed surfaces requires a substantial amount of post-processing time to complete using existing tools, with a significant portion spent on repetitive manual interaction and intervention.

Triangulation background

A Delaunay triangulation (e.g., Preparata 1985) is used in most geomatics software to produce Triangulated Irregular Networks (TIN) for data that can be projected to a planar surface (typically the XY or Easting-Northing plane). The algorithm maximizes the smallest internal angle (within that plane) to avoid skinny triangles and produces triangulations free of intersections by ensuring that the vertices of adjacent triangles lie outside of the circum-circle of each triangle. This technique produces excellent results on gently sloping topography; however, it produces "spiky" triangles when dense data points are obtained on near-vertical (*Z* direction or elevation) surfaces because the triangulation is determined by the *XY* (or East-North) distribution of the points, independent of the *Z* variation. Bonnaffe *et al.* (2007) and some software packages enable the user to modify the look direction of the Delaunay triangulation for better surfaces on cliff faces, but are still limited to a single plane of reference. Further, substantial user interaction is required to model sea caves, overhangs, and other spatially complex features common to coastal cliffs.

Many techniques exploit scans that are collected on a grid (spherical or cylindrical), where each scan is triangulated individually by connecting adjacent grid points and the results are subsequently merged into a single mesh. A distance threshold can be used to limit triangle sizes to reduce connecting grid points that have a large range gap. The zippered-mesh technique (Turk and Levoy, 1994) clips each scan by adjacent scans and then zips the adjacent scan meshes together. This technique is well-established for generating 3D models of objects, which can be scanned incrementally, with minimal overlap and occlusions. Unfortunately, to efficiently and adequately fill in data gaps from multiple occlusions found with complex topography, scans cannot be truncated, or large areas of the resulting mesh will be missing data. Particularly in topography, scans must be intermixed to produce a representative model without occlusions (*e.g.,* Olsen *et al.,* 2011). Also, an inherent problem with merging multiple scan triangulations with overlap is the potential for intersections, which are difficult to programmatically resolve. These intersections often must be manually repaired, which may introduce subjective selection.

Advancing front re-meshing techniques (Medeiros *et al.,* 2003; Sheng *et al.,* 2009) can be used to merge (or fuse) multiple surfaces together. When applying this method to seacliff modeling, the beach and cliff face are triangulated separately from different views and then fused together into a single model. The *advancing front* process starts with a seed triangle and grows outward along the point cloud forming triangles with the specified target triangle size. The *advancing front* process smoothes the data, particularly in areas where the individual meshes intersect.

The *ball and pivot* method (Bernardini *et al.,* 1999) has been successfully used for many cultural heritage applications (*e.g.,* Bernardini *et al.,* 2002). This method starts with a seed triangle and rolls a ball around on the edges of the triangle and forms triangles when the ball intersects with neighboring points. On generally smooth surfaces with limited scanner noise, this method can produce nearly clean surface meshes. Unfortunately, on complex topographical surfaces with sharp edges and with noisy scan data, the method can result in a substantial amount of intersecting triangles, which must be manually removed. Multiple passes are also required for variably spaced data, which is often the case in topographic scanning.

Scope

This paper presents an efficient method to produce a reliable surface mesh from large, noisy, 3D, point cloud datasets to enable modeling of complex topography while only requiring simple, intuitive user input parameters of data orientation, size of holes to fill, and the desired triangle size. The method focuses on triangulating all overlapping scans within a spatial region simultaneously to minimize inconsistent normals, intersections, and holes with minimal user interaction. Existing methods require substantial manual interaction and manipulation to produce a satisfactory mesh.

Methodology

Figure 1 shows an overview of the developed workflow of this triangulation method to generate surface models of seacliffs and beaches. The methodology was implemented and automated via creation of C++ code with a Graphical User Interface (GUI). Individual steps will be discussed in detail below:

User interaction and data preparation (Steps 1-2)

First, unwanted points need to be removed from the dataset. For example, noise (points) introduced by people walking in front of the scanner are generally manually eliminated, regardless of the meshing technique used. However, some statistical and image-based (Olsen *et al.*, In Press) filters can remove some of these artifacts but should always be examined for effectiveness.

Next, the user establishes the geometric properties (Figure 2) of the dataset including (a) the spatial extents to triangulate, (b) predominant orientation (determined by selecting two points), (c) cutoff elevation (Z_h, location of hinge between cliff and beach), and (d) cell dimensions ($\Delta X'$, $\Delta Y'$, $\Delta Z'$) based on the level of detail required for the model and data density.

Grid cell structuring and centroid filtering (Steps 3-5)

Following the user input, the rest of the workflow proceeds automatically. Each point in the dataset is binned into a cell (Figure 2), as determined by its position. Cells above the cutoff elevation are created in the Y'-Z' plane to represent the cliffs, and cells below are created in the X'-Y' plane to represent the beach. The centroid of all points binned into each cell is then calculated. This centroid approach has several advantages: (1) reduces redundant data, (2) produces a more uniform point cloud distribution, (3) removes some noise, and (4) reduces the effects of small offsets created from slight misalignments between overlapping scans on the resulting triangulation. Although the centroids are located in grid cells with fixed intervals, the actual centroid data remains as a pseudo-grid because the point spacings are not forced to constant intervals. This flexibility helps prevent blocky artifacts, resulting in improved, smoother topographic models.

Because the centroids are linked through pseudo-grid structures representing the cliff and beach, the triangulation of the data is simple and straightforward to construct using the rules established in Figure 2. Further, this data structure also means that additional memory does not need to be allocated to store the indices for the vertices of each triangle, as would be required with common triangulation techniques. It should be noted that the grid setup and centroid calculations are the only pre-processing steps required and the triangulation itself can be performed in real-time.

Grid Hole filling (Step 6)

Grid cells without a representative data point are filled by either linear or spline interpolation through points on a cross section. For the purposes of the results presented in this paper, linear interpolation was used because of improved efficiency with preservation of precision. Further, spline interpolations created abnormalities in the final model, unless high tension was used, which produces essentially the same results as linear interpolation at higher computation costs. The user can also select the size of holes to fill to avoid excess interpolation in large data gaps.

Cliff Triangulation (Step 7)

The triangulation algorithm marches through all rows and columns of the cliff grid using a 2x2 sub-grid. Because the centroid points are not fixed to a true, equally spaced grid, intersections can occur during triangulation, depending on the geometric arrangement of the points in four adjacent grid cells. Figure 3 shows the creation of two triangles from four points in adjacent grid cells to prevent intersections. If a convex hull (boundary) surrounding the points is a triangle, then the triangulation needs to be modified to account for the fact that one of the points lies inside of the triangle. Although four such scenarios exist, in addition to the most common scenario of a quad convex hull, the ordering of the points to form two triangles can be done only two ways. Thus, only two of the five scenarios (C and E) actually need to be tested when determining the triangulation sequence. An efficient point-in-triangle test (Akenine-Moller and Haines, 2002) can be performed by evaluating a points' barycentric coordinates (*i.e.,* area coordinates for a normalized triangle).

Beach Triangulation and Hinge Hole Filling (Steps 8-10)

Before the triangulation is performed on the beach grid, the indices of the bottom row of the cliff grid are copied to the beach grid (Figure 4), based on its X' coordinate, to stitch the beach and cliff data along the hinge. However, because the basic triangulation is performed for four grid points at a time, some holes can remain along the hinge. Figure 4 illustrates the procedure used to fill these holes. In this step, the algorithm compares the copied points between each adjacent row and pivots to formulate the necessary triangles to fill in the gaps.

Texture Mapping

Texture mapping is performed during steps 7, 9, and 10. Although not necessary for the geometric accuracy of the datasets and computation, the structuring of this approach enables 2D color images to be created for the cliff and for the beach to texture map the final model using the RGB values previously mapped to the point cloud. The resolution of the exported images is dependent on the selected cell size. If higher resolution imagery is desired, the output texture images can be up-sampled and the higher resolution images can then easily be mapped to the grid images.

Divide and Conquer optimization for long segments

In the case of very long segments, a divide and conquer procedure can be used where the datasets are split into smaller segments to reduce memory consumption and improve speed. To avoid gaps between sections, one row or column of points along the edge should be consistent between neighboring meshes.

Results and Discussion

Triangulation creation

Table 1 shows a comparison of the *hinged* method to two common types of triangulations, *advancing front* and *ball and pivot*. All processing was performed using an Intel Core2 Duo 6700@2.66 GHz system with 4GB RAM. The dataset consists of 16,893,325 points from approximately 40 terrestrial LIDAR scans performed along a 1.5 km section of coastline in Torrey Pines State Reserve, CA. In order to implement the *advancing front* and *ball and pivot* methods, the previously described centroid filter was used (0.25 m cell size) to produce a smaller dataset of approximately 2 million points. This filtering significantly reduced the amount of potential intersections by filtering out noise in the dataset. For the *hinged* triangulations, this step was performed as part of the processing calculations, as described previously.

Time for the creation of the mesh was divided into categories including: selection of input parameters, intermediate editing, triangulation creation time, and time required to clean the mesh. All methods require initial editing of the data to remove unwanted objects (*e.g.,* people, vehicles, animals, etc.), and thus that time (approximately 30 min) was not included in the comparison. The input parameters vary by method, with all requiring the input of a target triangle size. The *ball and pivot* and *advancing front* methods require the selection of a seed triangle or location. The *hinged* method requires the input of an elevation for the hinge and may require an azimuth rotation determined by the selection of two appropriate points. Only the *advancing front* method requires intermediate editing due to the creation of several initial surfaces that need to be edited prior to merging into the *advancing front* surface. The triangulation creation time is defined as the time required by the algorithms to structure the data and triangulate the points. It is the only time measurement that does not require interaction by the user. Finally, after the triangulation is generated, user time is required to remove any inconsistent facet normals or intersections and fill any holes that were not filled with the algorithm.

As shown in Table 1, all methods are simple and quick to implement. However, the *hinged* method provides substantial time savings (<4 min) when compared to the *advancing front* method (140 min) and the *ball and pivot* method (306 min) for satisfactory triangulation creation. The actual triangulation for the *hinged* method was done in real time (<<1 sec), so the 3.75 minutes includes data loading (~1 min), filtering (~1 min), hole filling (~30 sec), boundary hole filling (~15 sec), and data export to a file (~1 min). Because of no inconsistent facet normals, minimal holes and intersections, the *hinged* method required minimal time (<10 min) to manually edit, resulting in an acceptable surface. The *ball and pivot* method required the most time because of the substantial amount of intersections that needed to be resolved.

As shown in Table 1, the *advancing front* method can create intersections and inconsistent facet normals when merging multiple triangulations because of difficulties in resolving mesh direction in areas of overlap. These irregularities occur because as the *advancing front* surface grows outward, facets created facing the wrong direction are created in areas of complex topography or where there are differences between two intersecting surfaces. The *ball and pivot* method showed consistent normals throughout the surface. However, this method had substantially more intersections on the surface than the other methods, mainly because it was not designed for topography data with high rugosity. The *ball and pivot* surface contained the least amount of triangles.

It should be noted that the *hinged* method requires a specialized geometric configuration and orientation, while the other two methods can handle a more diverse range of geometries, making each respective method selectively advantageous. However, to counteract this, the *hinged* method can still be performed on specific faces and then fused with other meshes.

The *hinged* method, with and without the border hole filling procedure (Step 10), was also compared. Running the algorithm without border hole filling produced less intersections, but required many more holes to be filled. One advantage is that hole filling, particularly on small holes, can be easily performed automatically in many LIDAR software packages, while intersections must be removed manually. It should be noted that all of the intersections and holes occurred at the southernmost end of the dataset, where the cliff geometry immediately switches orientation from N-S (Y') to E-W (X'). Thus, when the cliff section modeled runs predominately in the Y' direction without abrupt shifts to the X' direction, the method produces meshes free of intersections and holes.

The final RMS residuals between the original ~17 million point dataset compared to the final models were calculated. The *advancing front* method had the lowest RMS residuals (3.2 cm) compared to the *hinged* method (3.8 cm) and the *ball and pivot* method (4.5 cm). It should be noted that the scanner used to collect the data has a nominal precision of 5 cm (1-σ) for field conditions within a range of 5-500 m; thus the residual results are very similar and reasonable for all of the methods evaluated.

Conclusions

The triangulation method presented herein produces continuous surfaces free of inconsistent normals, minimizing the occurrence of holes, and with minimal self-intersections, all of which reduce editing and production time. The generated meshes reduce noise in the data through statistical filtering and show good agreement with the original data points. The data structure of this method also enables several display optimizations to improve interactivity, while preserving data integrity. The resulting meshes are useful for volumetric change analysis and for general visualization purposes, particularly when modeling complex coastal features, such as overhangs and caves. These meshes were also similar in quality to those obtained by other techniques.

Acknowledgements

This material was supported by California Seagrant (Project #R/OE-39), the Coastal Environmental Quality Initiative (CEQI #04-T-CEQI-06-0046), and the National Science Foundation (#0403433). Leica Geosystems and Maptek I-Site generously provided software used in this study to aid in point cloud processing and triangulation evaluation. Jessica Raymond, Pat Rentz, and Jillian Maloney assisted with the survey work. Neal Driscoll and Scott Ashford contributed to this study. MeshLab, a tool developed with the support of the 3D-CoForm project, and the GLC library (developed by Laurent Ribon) were used for this study. The authors thank the reviewers for their comments.

References

Akenine-moller, T., and Haines, E. (2002). *Real-time rendering,* A.K. Peters, Ltd., Natick, MA, 578-572.

Bernardini ,F., Mittleman, J., Rushmeier, H., Silva, C., and Taubin, G. (1999). "The Ball-Pivoting Algorithm for Surface Reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, 5(4), 349-359.

Bernardini ,F., Martin, I., Mittleman, J., Rushmeier, H., and Taubin, G. (2002). "Building a Digital Model of Michelangelo's Florentine Pieta." *IEEE Computer Graphics & Applications*, 22(1), 59-67.

Bonnaffe, F., Jennette, D., and Andrews, J. (2007). "A method for acquiring and processing ground-based lidar data in difficult to access outcrops for use in three-dimensional virtual-reality models." *Geosphere*, 3(6), 501-510.

Medeiros, E., Velho, L., Lopes, H., (2003) "A Topological Framework for Advancing Front Triangulation," XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'03), 45.

Olsen, M.J., Ponto, K., Kimball, J., Kuester, F., & Seracini, M. (In Press). "2D open-source editing techniques for 3D laser scans," *Proceedings of Computer Applications and Quantitative Methods in Archaeology - CAA'2010*. Editors: Javier Melero, Pedro Cano, & Jorge Revelles.

Olsen, M.J., Johnstone, E., Kuester, F., Ashford, S.A., & Driscoll, N. (2011). “New automated point-cloud alignment for ground based LIDAR data of long coastal sections,” *J. Surv. Eng.,* 137(1), 14-25.

Olsen, M.J., Johnstone, E., Driscoll, N., Ashford, S.A., & Kuester, F., (2009). “Terrestrial laser scanning of extended cliff sections in dynamic environments: a parameter analysis,” *J. Surv. Eng.,* 135(4), 161-169.

Preparata, F. R., and Shamos, M. I. (1985). *Computational geometry: An introduction*, Springer, New York.

Sheng, L., Li-qu, L., and Xiao-main, C., (2009) "A New Mesh Growing Surface Reconstruction Algorithm," *First International Workshop on Education Technology and Computer Science (ETCS)*, 3, 889-893.

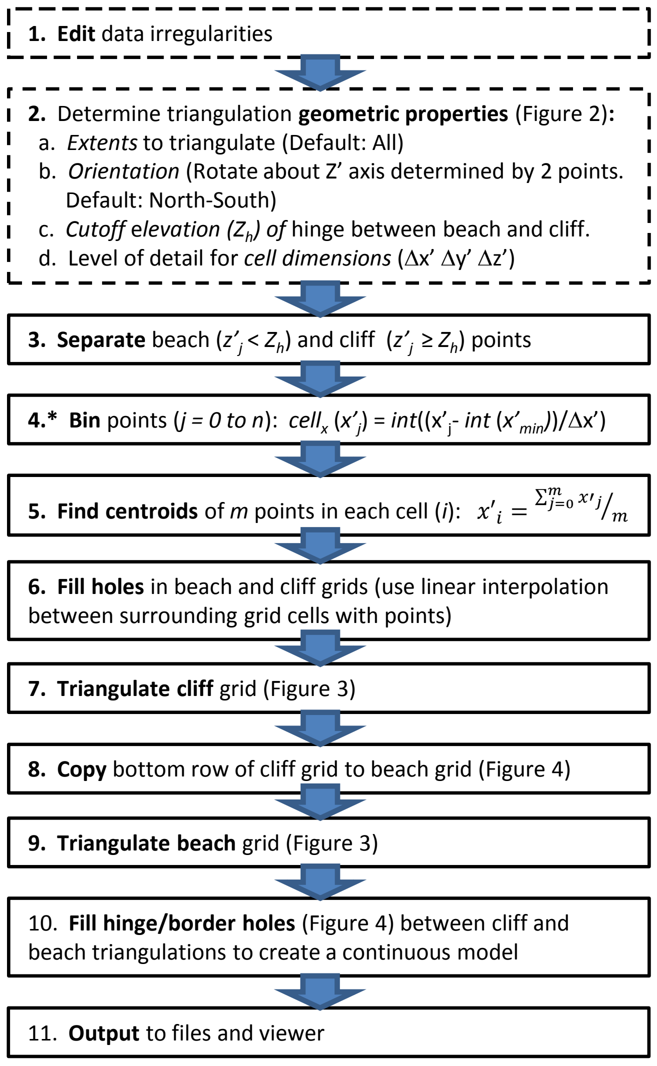
Turk, G. and Levoy, M. (1994). *Zippered Polygon Meshes from Range Images*. *Proc. ACM Computer Graphics, SIGGRAPH*, Orlando, Florida, 311-318.

Tables

Table 1. Comparison of time and quality for triangulation methods.
*** = user interaction required during process**

Triangulation Creation Time	**Advancing Front Method (using filtered data)**	**Ball and Pivot Method (using filtered data)**	**Hinged Triangulation**	**Hinged Triangulation (no border hole filling)**
1. Time to determine and enter Input parameters (min)*	3	**0.5**	1	1
2. Intermediate editing (min)*	30	**0**	**0**	**0**
3. Automated pre-processing and triangulation generation (min)	140	306	3.75	**3.25**
Triangulation Results (Before Final Editing)				
# Triangles	4,124,648	**2,062,761**	2,619,236	2,617,208
# Intersections	92	197,289	13	**8**
# Inconsistent normals	58	**0**	**0**	**0**
# internal boundary edges	41,170	150,602	**6**	2,725
Final Results (After user editing)				
4. Editing time to produce surface without intersections, holes or inconsistent normals (min)*	120	1030	**8**	10
5. Total time (min)	293	1336.5	**12.75**	14.25
Final RMS (cm)	**3.2**	4.5	3.8	3.8

Figures



1. **Edit** data irregularities

2. Determine triangulation **geometric properties** (Figure 2):
   a. *Extents* to triangulate (Default: All)
   b. *Orientation* (Rotate about Z' axis determined by 2 points. Default: North-South)
   c. *Cutoff elevation* (Z_h) *of* hinge between beach and cliff.
   d. Level of detail for *cell dimensions* ($\Delta x'$ $\Delta y'$ $\Delta z'$)

3. **Separate** beach ($z'_j < Z_h$) and cliff ($z'_j \geq Z_h$) points

4.* **Bin** points ($j = 0$ *to* n): $cell_x\,(x'_j) = int((x'_j - int\,(x'_{min}))/\Delta x')$

5. **Find centroids** of *m* points in each cell (*i*): $x'_i = {\sum_{j=0}^{m} x'_j}/{m}$

6. **Fill holes** in beach and cliff grids (use linear interpolation between surrounding grid cells with points)

7. **Triangulate cliff** grid (Figure 3)

8. **Copy** bottom row of cliff grid to beach grid (Figure 4)

9. **Triangulate beach** grid (Figure 3)

10. **Fill hinge/border holes** (Figure 4) between cliff and beach triangulations to create a continuous model

11. **Output** to files and viewer

* *int* rounds the value down and keeps only the integer portion.

Figure 1. Workflow for integrated seacliff and beach triangulation. Steps requiring user-interaction are shown with dashed boxes. Note that for simplicity, only calculations are shown for X' values. The same procedure is followed for Y' and Z' values.

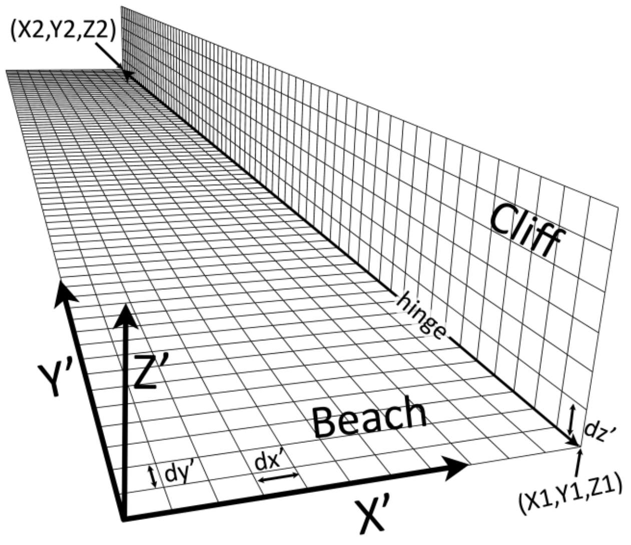


Figure 2 - Cube subdivision grids for cliff and beach with points used to determine rotation for cube coordinate system.

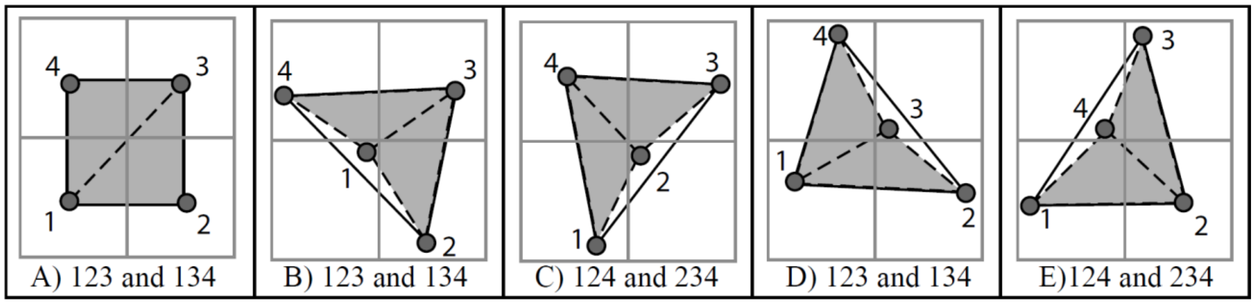
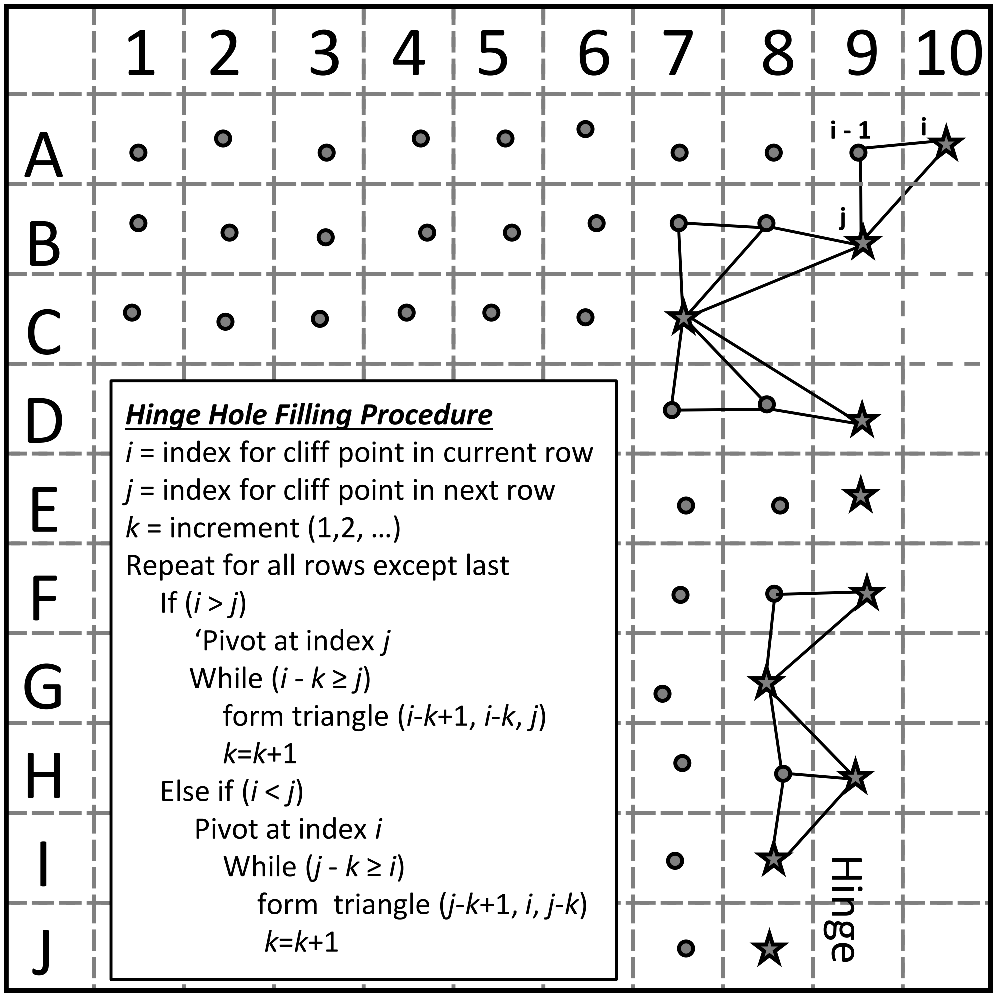


Figure 3- Determination of triangulation scheme for four grid cells. The convex hull is shown with the dark lines and the created triangles are shown with dashed lines.



Hinge Hole Filling Procedure

```
i = index for cliff point in current row
j = index for cliff point in next row
k = increment (1,2, …)
Repeat for all rows except last
    If ($i > j$)
        ‘Pivot at index j
        While ($i - k \geq j$)
            form triangle ($i-k+1$, $i-k$, $j$)
            $k=k+1$
    Else if ($i < j$)
        Pivot at index i
        While ($j - k \geq i$)
            form  triangle ($j-k+1$, $i$, $j-k$)
            $k=k+1$
```

Figure 4- Hinge/border hole filling procedure. The bottom row of the cliff grid (stars) is copied to the beach grid (circles). After the initial beach triangulation, these special case holes are filled in with new triangles (solid lines).