Proceedings of the 6th AGILE
April 24th-26th, 2003 – Lyon, France

STIN METHOD: SURFACE TIN REPRESENTATION BY DELAUNAY TENS CONSTRAINED BY OBSERVATION LINES

Edward Verbree, Peter van Oosterom

Department of Geodesy - GIS-technology

Delft University of Technology

Thijsseweg 11, 2629JA Delft

e.verbree@geo.tudelft.nl; oosterom@geo.tudelft.nl

1. INTRODUCTION

A proper representation of the surface of the Earth and what is build upon is needed as a data source for environmental modelling and planning. Especially Virtual and Augmented Reality applications require an appropriate representation of the actual terrain and man-made objects. One way to represent the terrain given by a set of surface points is to construct a Delaunay Triangular Irregular Network (DTIN). This DTIN is believed to give the 'best' triangular tessellation as the Delaunay empty circle criterion opts for well-formed 'fat' triangles and the resulting triangulation maximizes the smallest angle [1]. This idea is true for many applications, but it is not for visual and analytical queries dependent on the height of the surface. This limitation is given by the fact that the distribution of the triangular mesh is defined in the two-dimensional XY-plane and the Z-value of the surface points is not taken into account by the Delaunay empty circle criterion at all. Alternatively, Data Dependent Triangulations (DDTINs) aim to identify which triangulation over a given set of points will optimize some quality, i.e. the minimal spatial area of the surface or the volume below the resulting surface. The Z-value of the surface points is now taken into account, but still no certainty that the derived TIN represents the actual surface can be given. Hence, the reconstruction of the surface given by only the set of surface points is not unambiguous.

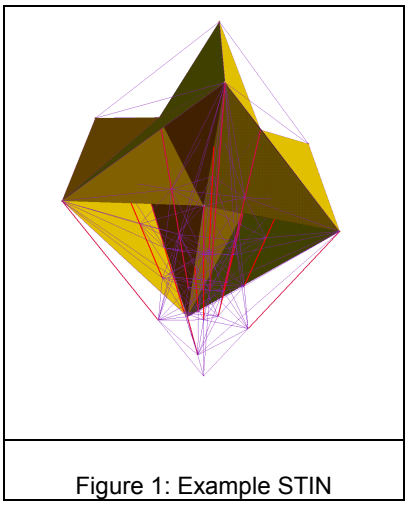

Figure 1: Example STIN

This paper describes a surface reconstruction method based on the Delaunay Tetrahedronised Irregular Network (DTEN), which tessellates the 3D-space with non-overlapping, adjacent, tetrahedrons. The DTEN is constructed by the Delaunay criterion, resulting in a tessellation where the circumscribing sphere of each tetrahedron is empty. The approach presented in this paper is new in that not only the surface points are included into the DTEN, but also the observation lines, i.e. the lines-of-sight between the observer (the measurement platform) and the target (the measured point), see figure 1. These observation lines constrain the DTEN such that the additional information is given to extract the Surface

TIN (STIN) from this DTEN. Afterwards the observation lines are discarded and they act for that purpose as a catalyst. Therefore the observation lines can also be artificial or simulated for this purpose. The STIN approach presented in this paper is an extension and refinement of the research presented in [2].

2. DELAUNAY TRIANGULATIONS AND DATA DEPENDENT TRIANGULATIONS FOR SURFACES

TINs are commonly used for surface representation. Given a dataset with target points on the surface and in addition breaklines and contourlines an Irregular Network of Triangles is created. The Z-value of these features is stored as the Z-value of the nodes of the computed TIN. A Delaunay TIN fulfils the 'empty circle criterion'. This criterion opts for the triangulation with 'fat' triangles, such that the triangulation maximizes the smallest angle.

Most commercial GISs have implemented this Delaunay TIN. We have to realise however that the 'empty circle criterion' does not take the Z-value of the features into account at all. This is clearly seen if the point distribution is square, as in the following example (figures 2a and 2b). In this figures 25 Target points are given, with an alternating Z-value of 1 or 2.

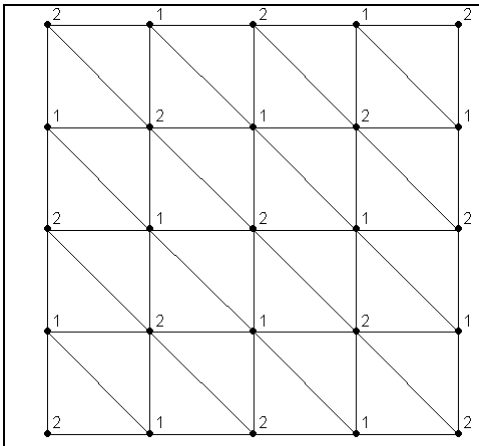

Figure 2a: One possible Delaunay TIN

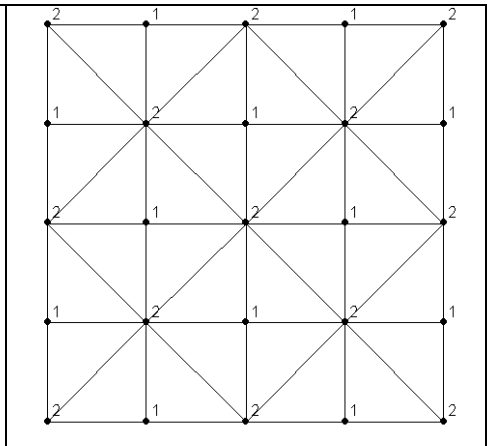

Figure 2b: Another possible Delaunay TIN

In figure 2a the diagonal of all triangles is directed northwest to southeast. This direction could be, with the same Delaunay criterion in mind, northeast to southwest for all diagonals or distributed as in figure 2b. Which one to choose?

The height values of the target points (or the Z-values of the nodes in the DTIN) do have consequences for derivatives like slope and aspect, visualization (hill-shading) and volume statistics (view sheds, and cut and fill calculations). One can argue that the 2D-Delaunay TIN (the triangulation of a 'flat' surface) is just one of the possibilities to triangulate a set of points and lines. In fact, any triangulation can be a candidate for a 2.5D terrain surface representation.

A better approach is to take the Z-value of the target points into account in the triangulation process. Extensive research on Data Dependent Triangulations (DDTINs) proves this observation. The idea behind this concept is to maximize or to minimize some cost function that expresses certain local, regional or global properties of the resulting surface [3,4]. Possible options for this cost functions are: minimize the surface area,

minimize the volume, minimize the maximum angle of the surface triangles, etc. But these local or global criteria could disregard certain phenomena, like ridges and faults, and as the projection is still made to the XY-plane, no overhanging cliffs or other disturbances are possible. The Surface TIN approach based on Delaunay TENs could solve these problems.

3. SURFACE REPRESENTATION: THE 1.5D CASE

In exploring the problem to retrieve a 2.5D surface within a Tetrahedronised Irregular Network in 3D we explain this procedure first for the 1.5D situation. Here we want to retrieve a 1.5D surface within a Triangulated Irregular Network in 2D. This seems to be quite trivial to do, because we can order the target points on X-value. But as, for our goal, this is not straightforward in two dimensions, we have to use an algorithm, which will not take this ordering as a precondition.

The aim is to find the '2D-volume' and the '1.5D-surface' defined by a set of target points and observation lines. The observation lines are shortened to a given value above the most extreme height value, and in this case dropped as perpendiculars.

We will give the algorithm is pseudo-code:

Step 0: Initialisation

```
Read target points
Define observation lines
Create 'initial, empty' TIN
```

Step 1: Construct observation-line constrained TIN - see figure 3a

```
Add target points to TIN
Add observation lines as constrain to TIN
  Split observation-line at Steiner_Points until
    each observation-line is TIN_Edge
```

First a DTIN is created by a set of target points and observer-points. Then, an iterative process is started. Each observation-line not being a TIN_Edge is forced to subdivide into parts at a Steiner point. These Steiner points are included into the DTIN. This step finishes when all observation lines are represented by TIN_Edges in the Delaunay TIN. The addition of Steiner points is a powerful concept and is used in this approach as mean to find the Surface TIN.

Step 2: Transform TIN_Edges to Volume_Edges – see figure 3b

```
for each TIN_Edge
  if TIN_Edge has (Target_Point1, Steiner_Point)
    Find Target_Point2 at end of Steiner_Point's observation-line
    Replace in TIN_Edge Steiner_Point with Target_Point2
  end
  if TIN_Edge has (Target_Point1, Target_Point2) then
    Construct Volume_Edge (Target_Point1, Target_Point2)
    Add Edge to List_Volume_Edges
  end
```

```
end
```

The ‘2D Volume’ and the ‘1.5D Surface’ are found by the procedure, given by the code in step 2. All TIN_Edges are examined. If within a TIN_Edge a Steiner point is present the Target point at the end of the observation-line replaces this one. These newly constructed TIN_Edges are not Delaunay anymore, but are needed to obtain the 1.5D Surface and 2D Volume. All TIN_Edges (partly Delaunay, partly not) with two target points are maintained, the other TIN_Edges are discarded. The remaining TIN_Edges compose a complete and valid TIN.

Step 3: Find STIN_Edges on Surface by hidden_edge removal

```
for each Volume_Edge in List_Volume_Edges
  if Volume_Edge is 'below' any Edge in List_Volume_Edges
    add Volume_Edge to List_Removed_Edges
  else
    add Volume_Edge to List_STIN_Edges
  end
end
```

Finally a hidden line removal algorithm retrieves the Surface_Edges. A TIN_Edge is considered as ‘below’ another TIN_Edge if it has a target point in common and the Z-value of the mid of the TIN_Edge is less than the Z-value of the comparing TIN_Edge. In this 1.5D example the hidden line removal algorithm is quite trivial, which can be easily extended to a 2.5D hidden face removal, as all observation lines are dropped perpendicular to the Target_Points. For ‘real’ 3D cases a more demanding hidden face removal algorithm has to be applied, as the observer point could be anywhere in the scene.

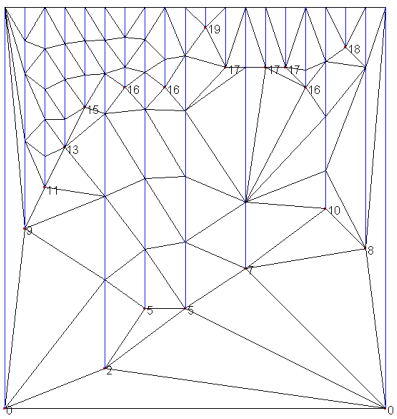

Figure 3a: Delaunay TIN constrained by observation lines

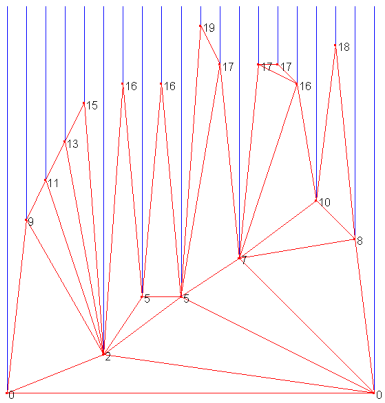

Figure 3b: Derived ‘2D-Volume’ and ‘1.5-Surface’

4. SURFACE REPRESENTATION: THE 2.5D CASE

The 2.5D case is not as trivial as the 1.5D example given in the previous paragraph. This 1.5D case was given to demonstrate the steps to be taken in the algorithm to construct

the Surface TIN (STIN), but it has no practical use. The STIN algorithm in 2.5D makes sense, because in this scenario the height value of the target points on the terrain surface can be considered as one possible attribute value of the planimetric X,Y co-ordinates. This limitation holds for most real-world terrains, with the exception of cliffs. Extending the STIN method for 3D Surfaces, which is part of research to undertake, can solve these kinds of situations.

As in the 1.5D scenario the observation of the 2.5D target points are dropped as perpendiculars. The target points and the observation lines are now included into a Delaunay Constrained Tetrahedronised Irregular Network. This 3D-Network should result in a set of non-overlapping, adjacent, tetrahedrons, which together fill a convex solid (volume). Each network of tetrahedrons should adhere to the following:

a: Of each tetrahedron its four vertices should not be located in the same plane.
b: Each tetrahedron should not contain any other points of the dataset.
c: A TEN-Face (triangle) is on the boundary of the solid or is exactly shared by two internal tetrahedrons.

To create a set of Delaunay tetrahedrons one condition has to be added:
d: For each of the tetrahedrons in a Delaunay TEN the circumsphere should not contain any other point of the dataset

To create a TEN constrained by observation lines the last condition taken into account is:
e: All observation lines are identified as edges in the Delaunay TEN.

We will apply the same algorithm as in the 1.5D scenario, but all operations are one dimension higher. However, some special difficulties have to be solved. We give the procedure again with some pseudo-code, but the example dataset showed in figures 4a - 4f gives also a good inside in the method. This dataset has a cluster of elevated points in the northwest, with a cluster of low points as neighbours. These clusters have to be remained in the derived Surface TIN (STIN).

The procedure is roughly as follows:

Step 0: Initialisation - see figure 4a

```
Read target points
Define observation lines
Create 'initial, empty' TEN
```

Step 1: Construct observation-line constrained TEN - see figure 4b

```
Add target points to TEN
Add observation lines as constrain to TEN
Split observation-line at Steiner_Points until each observation-line is TEN_Edge
```

Step 2: Transform TEN_Edges to Volume_Edges - see figure 4c

```
for each TEN_Face
  if TEN_Face has (Target_Point1, Target_Point2, Steiner_Point)
    Find Target_Point3 at end Steiner_Point's observation-line
    Replace in TEN_Face Steiner_Point with Target_Point3
  end
```

```
  if TEN_Face has (Target_Point1, Target_Point2, Target_Point3 then
    Construct Volume_Edge1 (Target_Point1, Target_Point2)
    add Edge1 to List_Volume_Edges
    Construct Volume_Edge2 (Target_Point2, Target_Point3)
    add Edge2 to list_Volume_Edges
    Construct Volume_Edge3 (Target_Point3, Target_Point1)
    add Edge3 to List_Volume_Edges
  end
end
```

Step 3a: Find STIN_Edges on Surface by hidden_edge removal - see figure 4d

```
for each Volume_Edge in List_Volume_Edges
  Find List_Intersected_Edges // Projected in 2D
  if Volume_Edge is 'below' any Edge in List_Intersected_Edges
    add Volume_Edge to List_Removed_Edges
  else
    add Volume_Edge to List_STIN_Edges
  end
end
```

Step 3b: Check Removed_Edges to include in STIN_Edges - see figure 4e

```
// Special Case: check List_Removed_Edges
// Non intersecting Removed_Edges are to be included in STIN
for each Volume_Edge in List_Removed_Edges
  if Volume_Edge has no intersection in List_Removed_Edges
    add Volume_Edge to List_STIN_Edges
  end
end
```

Step 4: Create STIN_faces - see figure 4f

```
for each STIN_Edge in List_STIN_Edges
  find List_Connected_Edges in List_STIN_Edges
  for each Connected_Edge in List_Connected_Edges
    if connected_Edge and STIN_Edges has one connected Node
      create STIN_Face (STIN_Edge, Node)
    end
  end
end
```

Step 0, 1 and 2 are not really different from the 1.5D Surface TIN approach. The crux is in steps 3a and 3b. In step 2 the Volume_Edges are found by examination of all TEN_Faces. If a TEN_Face has one Steiner point and two target points the Steiner point is replaced by the target point at the end of the observation-line. The TEN_Edges of this

TEN_Face are stored as Volume_Edges. Also the TEN_Edges of the TEN_Faces with three target points are stored as Volume_Edges.

In Step 3a a hidden edge removal algorithm is applied on the Volume_Edges to retrieve the STIN_Edges. First all Volume_Esges are declared as STIN_Edges. The algorithm applied projects each Volume_Edge to 2D and test this one to the projected and intersecting other Volume_Edges. The intersection point is calculated in 2D, and the algorithm continues with the Z-value calculation of the Volume_Edges at the intersection point. The Volume_Edge with the lowest Z-value is removed form the STIN_Edges and declared to be a Volume_Edge.

A problem arises in that some removed Volume_Edges are to be considered as STIN_Edges to obtain a complete and valid STIN_Faces. The removed Volume_Edges that has no 2D-intersection with another removed Volume_Edge are promoted to STIN_Edges. This check is performed in step 3b.

Finally the STIN_Faces are constructed in step 4. When this STIN_Faces are visualised in 2D as in figure 4f-beneath, it is clearly shown that some very thin triangles are formed. In most literature these thin triangles are considered as ‘bad’ and have to be avoid as a Delaunay triangulation does. But as stated in the introduction the STIN is to be considered as a Data Dependent Triangulation, which tries to fit a boundary representation given by a set of target points and has to be judged on volumetric or spatial area of the surface.

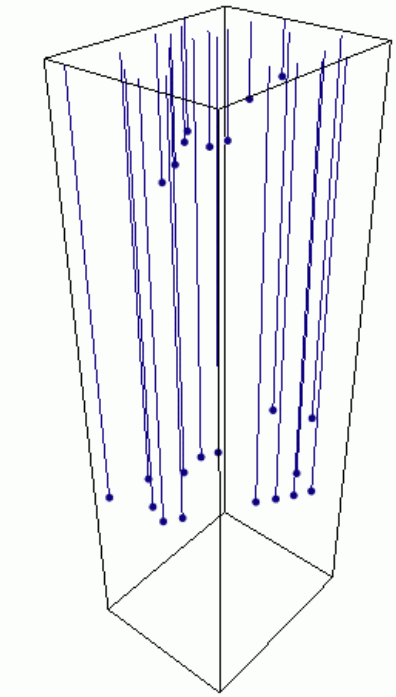

Figure 4a: Target points + Observation lines

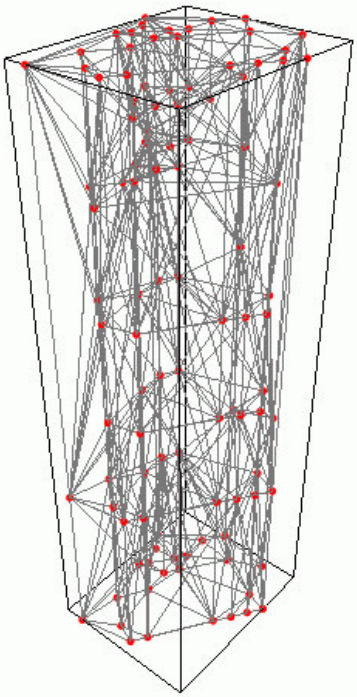

Figure 4b: Constrained TEN

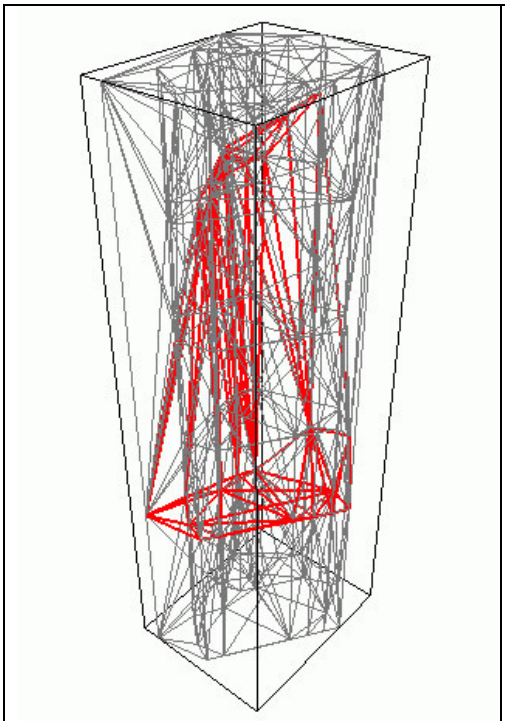

Figure 4c: Volume_Edges

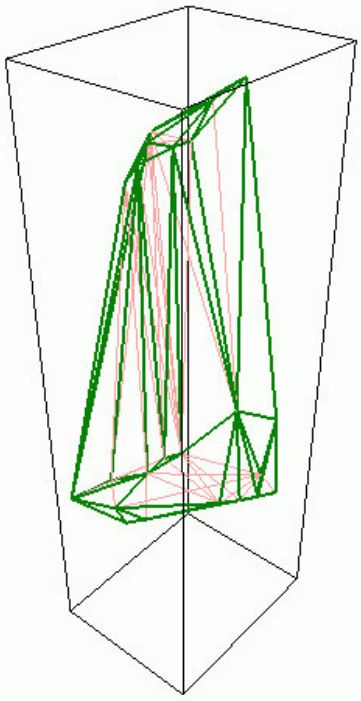

Figure 4d: Visible Surface Edges

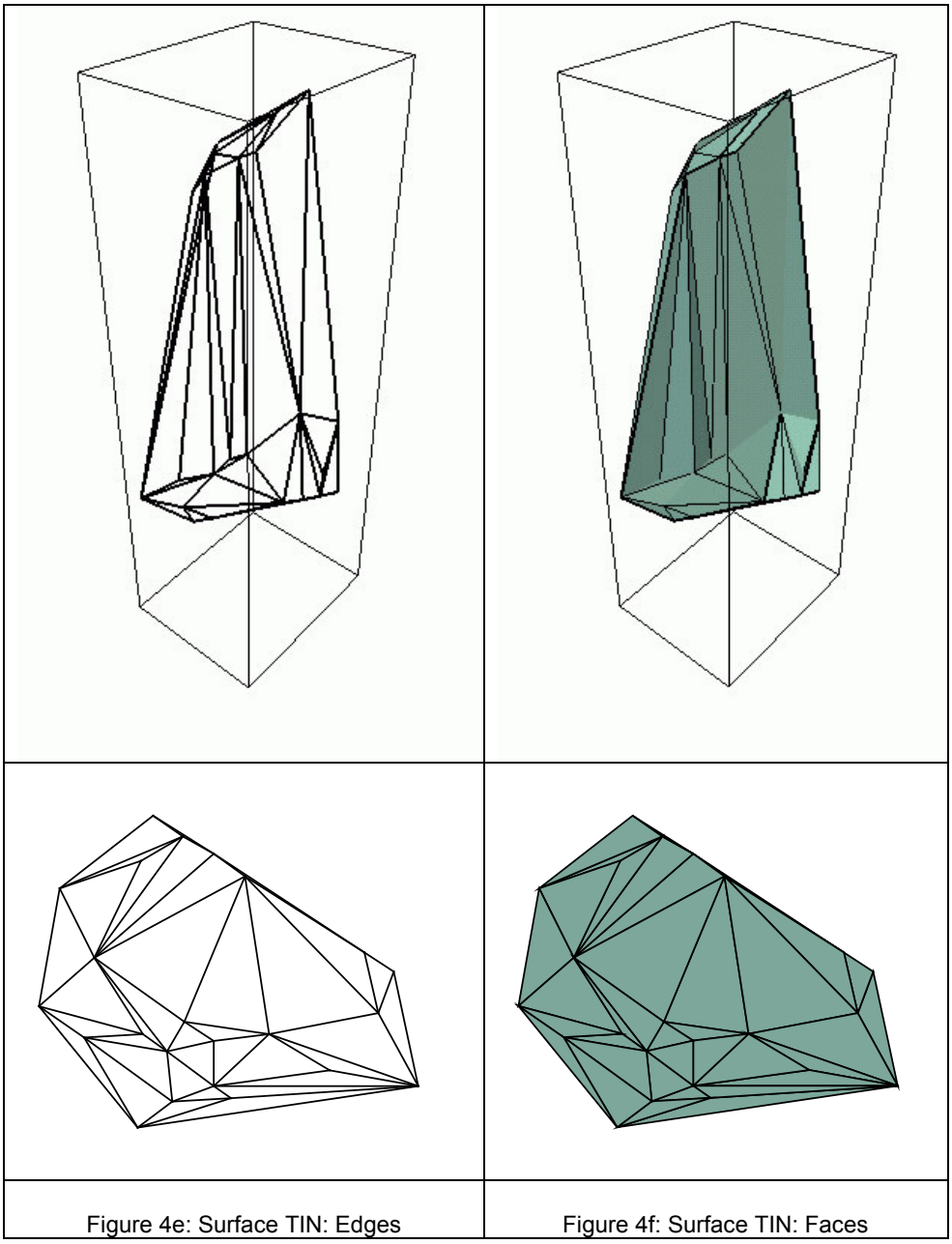

Figure 4e: Surface TIN: Edges

Figure 4f: Surface TIN: Faces

5. CONCLUSIONS AND CURRENT RESEARCH

The standard Delaunay TIN (DTIN) method has to be handled with care when used for height-dependent applications, as the Z-value of the target points is not considered in the construction. Within the Surface TIN (STIN) method the Z-value of the target points is taken into account when this surface is created and derived within a Tetrahedronised Irregular Network (TEN) in three dimensions. This method lines up with all kinds of Data Dependent

Triangulations (DDTIN). The surfaces created with the STIN method are to be examined in detail and compared to the results obtained with DTINs en DDTINs.

This STIN method is able to reconstruct 2.5D Surfaces. Current research is undertaken to extend the method for full 3D Surface reconstruction. The research involves:

- Extension of the STIN method for observation scans form multiple directions and locations to find a solution for overhanging cliffs and caves.
- To give a formal proof the resulting STIN is a valid TIN with no intersecting or missing edges.
- The combination 2.5D STIN terrain representation with 'true' 3D TEN objects
- Use of real world dataset to test the procedures for correctness and robustness.

6. REFERENCES

[1] C.L. Lawson. Software for C1 Surface Interpolation, in: Rive, J. (Ed.) Mathematical Software III, pp. 161-194.

[2] Edward Verbree en Peter van Oosterom, Scanline forced Delaunay TENs for surface representation, proceedings IAPRS, Volume XXXiV-3/W4, Annapolis, MD-USA, October 2001.

[3] N. Dyn, D. Levin and S. Rippa. Data dependent triangulations for piecewise linear interpolation. IMA J. Numer. Anal. 10 (1990), pp. 137-154.

[4] Ulrich Lenk, Optimisation Criteria for Degenerated Delaunay Triangulations, proceedings GIScience 2000, Savannah, Georgia, USA, October 28-21, 2000.