**GISC 6384 – Spatial Analysis and Modeling**

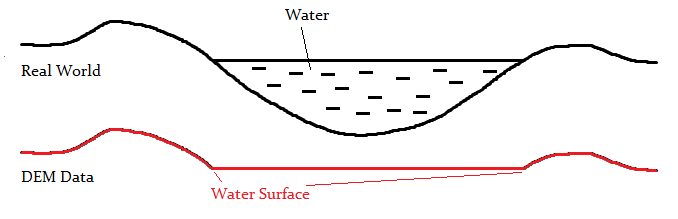
**Final Project Report**

**Water Surface & Sink Detection Based on DEM**

*By Caiyi Zhong*

1. **Introduction**
   1. **What are water surfaces?**

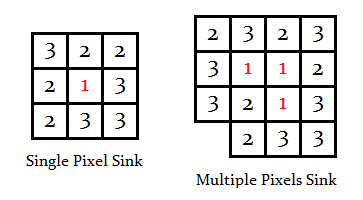
In some kinds of DEM dataset, a water surface would be like this:



So the water surface could be defined as a somewhat “flat” part in a DEM. Consider it as a lake. This can dramatically affect hydrology models such as Flow Direction, Sink, etc. So before we conduct hydrology analysis, we must take this into consideration.

* 1. **What are sinks?**

Simply said, a sink is a pixel whose neighbors (8-neighbor) are all higher than itself. This concept could also be extended to multiple pixel areas. If the area’s all neighbors is higher than itself, while all the pixels within this area has the same value of elevation, then this area could be regarded as a sink. That is to say: If a water surface is small enough, then it could become a sink.



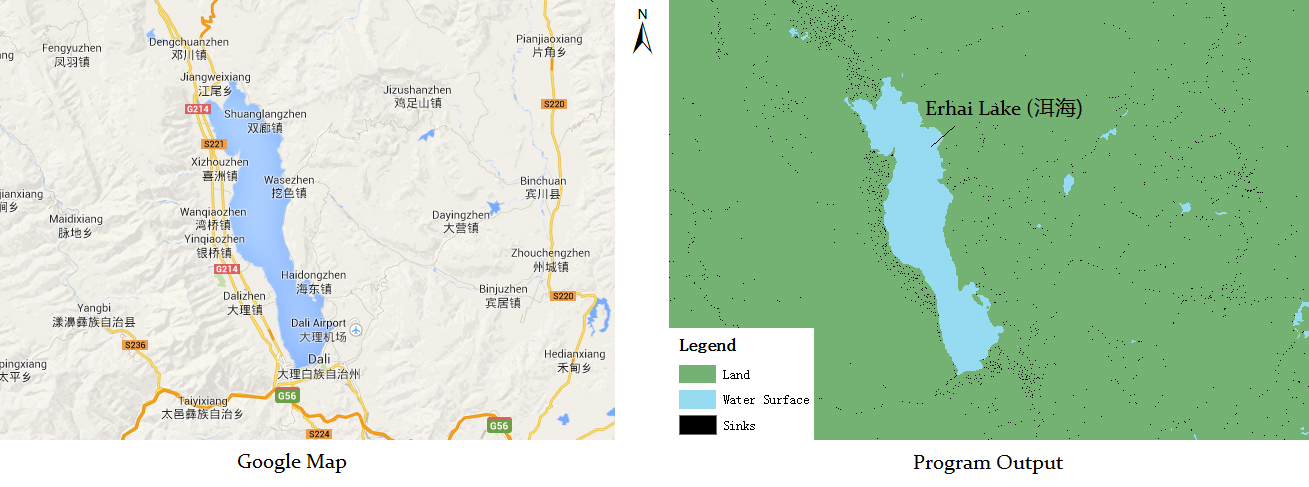
Typically we use Flow Direction grid to calculate sinks. But in this case, assuming our Flow Direction won’t work due to our water surface issues, sinks cannot be generated from this way. Fortunately, we can directly use the algorithm which can detects water surface, to find sinks.

* 1. **What would this solution do?**

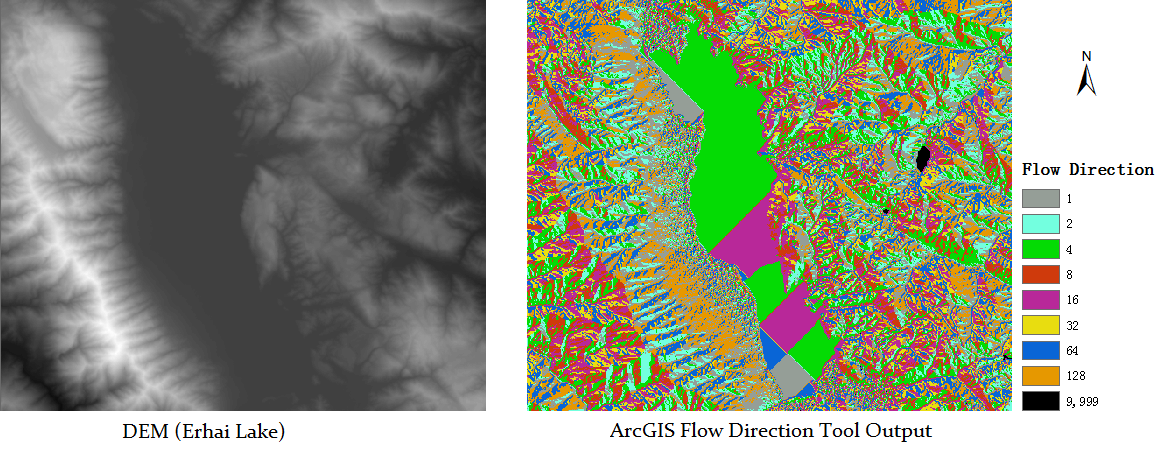
This program can generate an integer raster as the output. This raster has the same extent, cell size and spatial reference as the input grid, with each pixel assigned with a value 0, 1, 2 or 3. 1 for land, 2 for water and 3 for sinks. 0 is reserved for error pixels, and normally the output would not contain any 0 valued pixel.

Here is a sample data and result at Dali, Yunnan Province, China.

We can see The Erhai Lake has been detected distinctly.

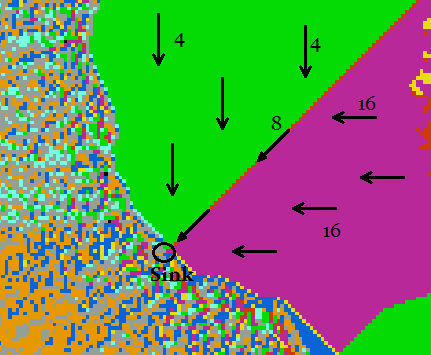


1. **Problem Statement**
   1. **How water surfaces affect Flow Direction analysis?**



From this we can see how extremely flat areas affect Flow Direction analysis. According to ESRI documents [Reference 1]: *If the maximum descent to several cells is the same, the neighborhood is enlarged until the steepest descent is found*. So if we want to calculate the flow direction of a pixel within a flat area, the neighborhood would be enlarged again and again, and eventually something like above would be generated.

Sinks would also appear at the bank of a lake, making the flow direction like this:



This would not be what we want.

* 1. **Objectives**

What we do here could be summarized as:

* + - * Detect water surfaces and sinks from a DEM before conduction Flow Direction analysis
      * Improve Flow Direction’s performance by removing water surface areas and sinks from consideration

Then we can get a better Flow Direction output to do further researches.

1. **Literature Review for Water Surface Detection**

Existing water surface detection studies are mostly based on multi-spectral remote sensing and LIDAR. Multi-spectral analysis is easy to understand, before doing this we must collect multi-spectral data first and use classification on different band compositions. LIDAR method is another good way to solve this problem, but here, In my case, we can only use DEM to do this.

1. **Data Sources, Tools and Environments**
   1. **Data Sources**

NASA 2003 DEM

Including China, 30 meters resolution

Downloaded from: <http://www.fengshui-168.com/thread-99833-1-1.html>

In this case, I use the data of Yunnan Province as a sample.

* 1. **Tools and Development Environment**

Programming Language:

C++

Library:

GDAL - Geospatial Data Abstraction Library

<http://www.gdal.org>

Programming Tools and Environment

Microsoft Visual Studio 2012

1. **Analysis and Methodology**
   1. **Overall Algorithm**

**Input:** DEM data raster

**Output:** An integer raster indicating the detecting result

Before we start, we define some macros to explain our output values:

#define WDRESULT\_UNKNOWN 0

#define WDRESULT\_LAND 1

#define WDRESULT\_WATER 2

#define WDRESULT\_SINK 3

We also need to check a pixel’s (px’s) neighborhood, so we define these possible check results:

enum WDNeighborCheck {

WDNEIGHBOR\_UNKNOWN, // Error or unchecked

WDNEIGHBOR\_CAN\_FLOW\_DOWN, // One or more neighbor lower than px

WDNEIGHBOR\_SINK, // All neighbors higher than px

WDNEIGHBOR\_HALF\_SINK, // Part of neighbors higher than px

WDNEIGHBOR\_FLAT // All neighbors equal to px

};

At first we create a 2-d buffer named resultGrid with the same width and height with the input DEM grid, and fill this grid with WDRESULT\_UNKNOWN, indicating all the pixels have not been processed. Then the algorithm could be explained with the pseudo code below:

for each non-processed px (i.e. pixel) in resultGrid

switch (*CheckNeighbor*(px))

case WDNEIGHBOR\_CAN\_FLOW\_DOWN:

px = WDRESULT\_LAND

case WDNEIGHBOR\_SINK:

px = WDRESULT\_SINK

case WDNEIGHBOR\_HALF\_SINK:

*Floodfill*(px, resultGrid)

default:

// Do nothing

end switch

end for

The function *Floodfill*(px, resultGrid) is just like the Paint Bucket tool in MS Paint. Here I use a quick algorithm called *Queue-Linear Flood Fill* by J. Dunlap [Reference 2]. All of the flood-filled pixels in this function would be flagged as processed in order to prevent from duplicate processing.

* 1. **Methodology**

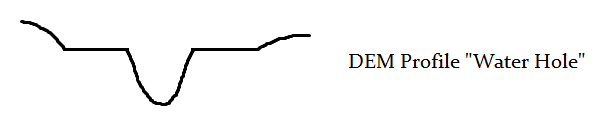
This algorithm is based on these assumptions:

* + - * Water surfaces is completely flat (allow some tolerance)
      * Water will flow down to lower area

For example, such area can be regarded as water surface:



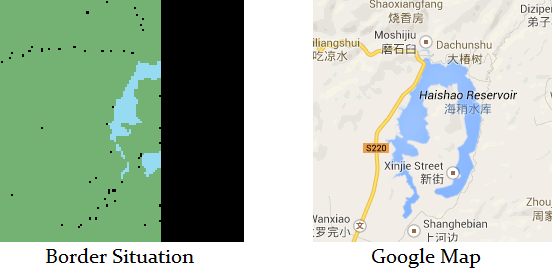
Waters cannot flow up, so such DEM pattern could be water area. But for the DEM pattern below:



Obviously, if the two segments of flat surface is water surfaces, the water must flow down into the hole in the middle. Therefore, such DEM pattern can only be regarded as land (maybe a sink in the hole). This should be straightforward.

* 1. **Border Situations**

For borders, we just simply assume that the border is infinite high, so that when a water surface is cut by a border, the result could keep correct.

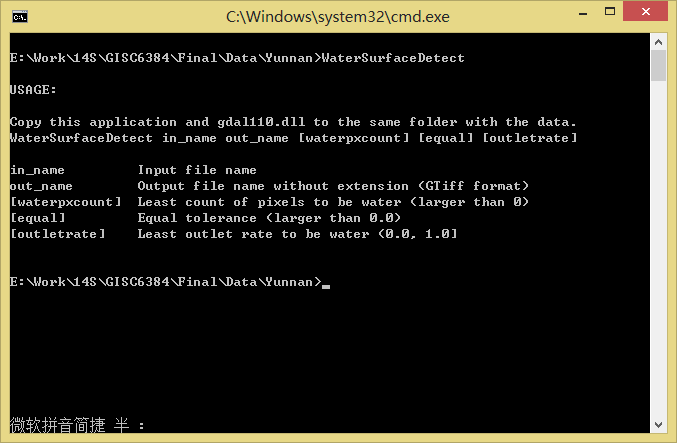


Here we can see the Haishao Reservoir is correctly detected.

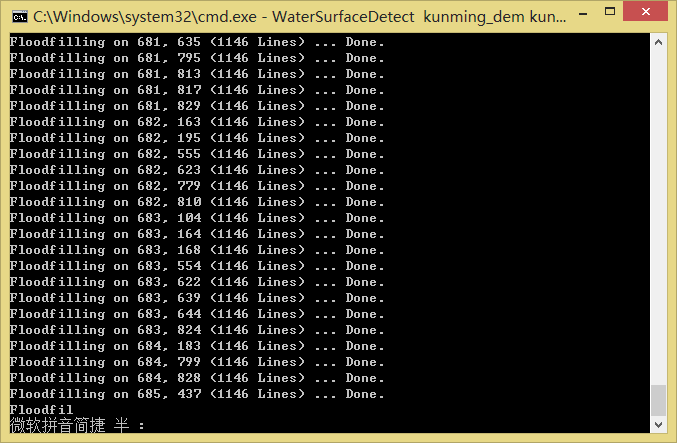
1. **Implementation**

The interface looks like this:

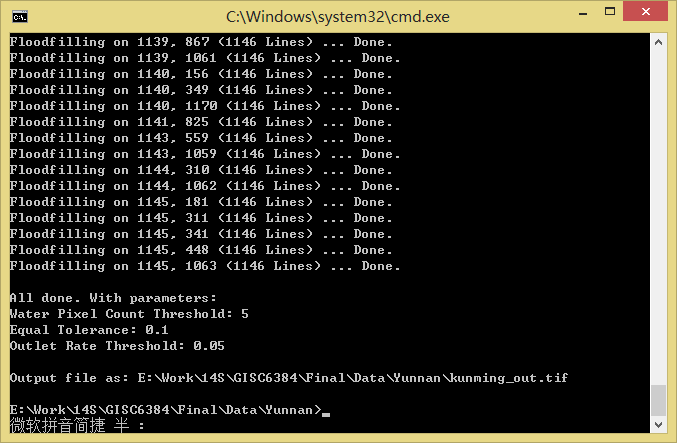
Enter the application name to get help, enter the application along with proper parameters to execute.



User Interface

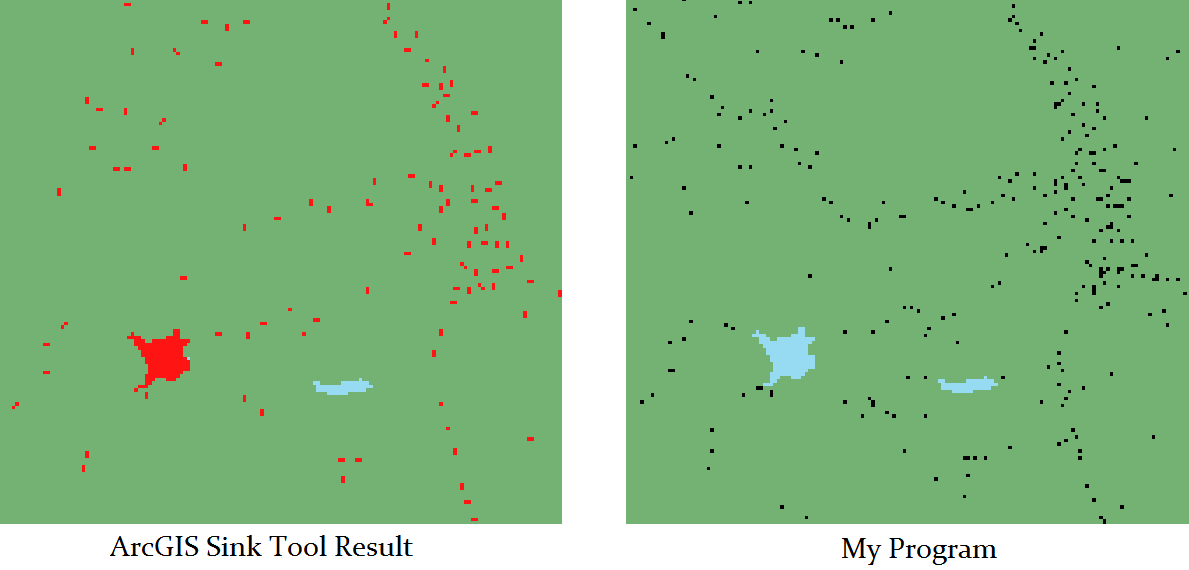


Running



Running success

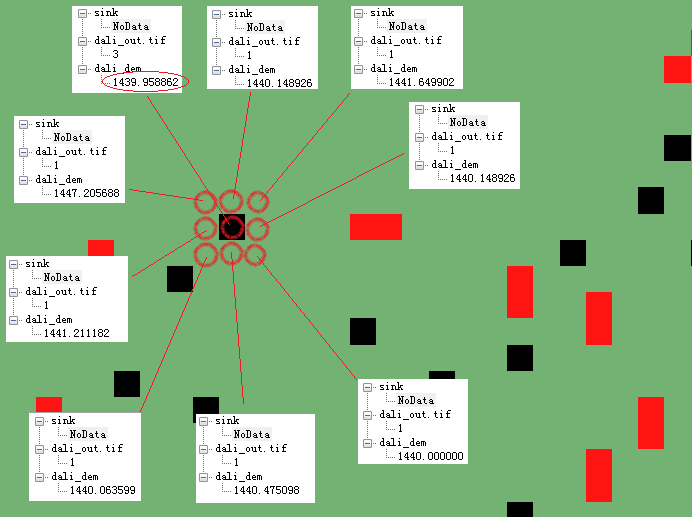
1. **Results and Discussions**
   1. Compared with ArcGIS’ Sink tool



Red dots are Sink tool output, while black dots are pixels assigned with WDRESULT\_SINK in my program. We can tell that black dots are more than red ones. What’s more, black ones covered all red ones, except a relatively big sink – actually it is a water reservoir, called *Dayindian Resorvoir*, according to Google Maps.



So is the black ones are really sinks, or just calculation errors? We can tell by using Identify tool of ArcMap:



Black dots are sinks MY PROGRAM GOT WHILE ARCGIS DIDN’T GET. I use Identify tool to check one black dot’s neighborhoods. The result is like above.

It shows that my program generates more correct sinks.

1. **References**

[1] How Flow Direction works, ESRI, ArcGIS Resources

<http://resources.arcgis.com/en/help/main/10.1/index.html#//009z00000063000000>

[2] Queue-Linear Flood Fill: A Fast Flood Fill Algorithm, J. Dunlap, Code Project

<http://www.codeproject.com/Articles/16405/Queue-Linear-Flood-Fill-A-Fast-Flood-Fill-Algorith>